University of Alberta

REAL-TIME DIGITAL SIMULATION OF LARGE POWER SYSTEMS BASED ON A ROBUST TWO-LAYER NETWORK EQUIVALENT

by

Xin Nie

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**

Department of Electrical and Computer Engineering

Edmonton, Alberta
Fall 2005

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-494-09247-5
Our file Notre référence
ISBN: 0-494-09247-5

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

In this thesis, a robust approach of Two-Layer Network Equivalent (TLNE) is elaborated. Featuring low-order Marti's line model in surface layer, global searching of low-order deep region networks by Genetic Algorithms (GAs) with passivity and stability constraints and compensation, and Constrained Linearized Least-Square (CLLSQ) optimization with guaranteed convergence as well as the stability and passivity of the obtained model, the robust approach is able to generate the Robust TLNE model with high accuracy and efficiency.

Simulation results and computational time analysis show that the Robust TLNE model is highly efficient in the order reduction of the external system and is suitable for real-time implementation of larger power systems. Two examples and a case study of a realistic large power system — the Alberta Interconnected Electric System (AIES) verify the proposed Robust TLNE approach.

Furthermore, real-time implementation of the second example and AIES is accomplished at 20μs time-step size in a Xeon cluster based real-time simulator in the Real-Time eXperimental LABoratory (RTX-LAB) of the Power Engineering Group at the University of Alberta. A MATLAB/SIMULINK C++ S-function implementing EMTP is programmed to accommodate the Robust TLNE model in the real-time simulator. Real-time simulation results observed from oscilloscope are identical compared to off-line results.

Acknowledgements

I wish to express my deep gratitude to Dr. Venkata Dinavahi, my supervisor, for his constant and kind support, excellent advice and guidance during the whole research work.

I extend my thanks to the members of my M. Sc. committee members for their careful reviews of the thesis and for many useful comments.

Appreciations are also due to Dr. Bjørn Gustavsen from SINTEF Energy Research Norway for providing Vector Fitting related routines and helpful suggestions; to Dr. Washington Neves for providing the nonlinear fitting routine for transmission line parameters and kind assistance; to the Ms. Pamela Mclean from Alberta Electric System Operator (AESO), Calgary for providing Transmission Alberta System Model (TASMo) database, Alberta Interconnected Electric System (AIES) Map and valuable advice; to European EMTP-ATP User Group (EEUG) for their passion on working with, developing and enhancing ATP; to technical support of Opal-RT Technologies Inc. during the integration, testing and application of the RTX-LAB real-time simulator.

I am wholly indebted to my parents for their support and wisdom that has brought me this far and will hopefully carry me further.

Contents

1 Introduction 1

1.1 Frequency-Dependent Network Equivalent 2
1.2 Two-Layer Network Equivalent 3
1.3 Existing Issues Associated with TLNE 5
1.4 Thesis Objectives 6
1.5 Thesis Outline 6

2 Robust Two-Layer Network Equivalent for Passive Networks 8

2.1 Models and Concepts in the Robust TLNE 8
2.1.1 Positive-Realness and Passivity Criterion 8
2.1.2 Frequency Scan of Linear Passive Networks 9
2.1.3 Vector Fitting 11
2.1.4 Electrical Network Realization 12
2.1.5 Frequency-Dependent Transmission Line Model 13
2.1.6 Genetic Algorithms 18
2.2 Surface Layer 21
2.3 Deep Region 23
2.4 Finding the First Approximations with GAs 24
2.4.1 Problems in Finding the First Approximations 24
2.4.2 Application of GAs 25
2.4.3 Data Preparations For GAs 25
2.4.4 GA Parameters and Flowchart 26
2.4.5 GA Objective Function 29
2.4.6 Compensation Technique for GAs 30
2.4.7 Building the First Approximation of Input Admittance 31

2.5 Constrained Linearized Least-Square Optimization 31
2.5.1 Parameters Subject to Optimization 31
2.5.2 Linearization 32
2.5.3 The Jacobian Matrix 33
2.5.4 Iterative Least-Square Optimization 38
2.5.5 Optimal Deep Region Order Determination 39
2.6 Example 1 40
2.6.1 Generation of the Robust TLNE 41
2.6.2 Transient Simulations 41
2.7 Summary 42

3 Robust Two-Layer Network Equivalent for Active Networks **46**
3.1 External System with Active Elements 46
3.1.1 The Analytical Method 47
3.1.2 Measuring Short-Circuit Current 48
3.2 Robust TLNE for Three-Phase Multi-Port Systems 49
3.2.1 Clarke's Transformation 49
3.2.2 Nonlinear Fitting Method for Transmission Line Parameters . . . 52
3.3 Example 2 53
3.3.1 Generation of the Robust TLNE 53
3.3.2 Transient Simulations 62
3.4 Summary 63

4 Case Study — the Alberta Interconnected Electric System **68**
4.1 The Alberta Interconnected Electric System 68
4.2 Transmission Administrator System Model of AIES 71
4.3 Modeling and Network Reduction of AIES 72
4.4 The Robust TLNE model for AIES 75
4.5 Transient Simulations 84
4.6 Summary 84

5 Real-Time Simulations Based on the Robust TLNE Model **86**
5.1 Opal-RT Real-Time Simulator at RTX-LAB 86

5.2 Implementation of EMTP in MATLAB/SIMULINK 89
5.2.1 C++ Implementation of EMTP 89
5.2.2 SIMULINK S-Function Program Implementing EMTP 99
5.2.3 Working with EMTP S-Function Block in Real-Time Simulator . . 100
5.3 Real-Time Simulation of Example 2 108
5.4 Real-Time Simulation of AIES 108
5.5 Summary 108

6 Conclusions and Future Work **113**

Bibliography **117**

Appendix A Example 1 **123**
A.1 System Parameters 123
A.2 System Diagram in ATPDraw 124
A.3 ATP Data File for Full Model 125
A.4 ATP Data File for Robust TLNE Model 126

Appendix B Example 2 **127**
B.1 System Parameters 127
B.2 System Diagram in ATPDraw 128
B.2.1 Capacitor $C1$ Switching Case 128
B.2.2 Balanced Three-Phase to Ground Fault Case 129
B.3 ATP Data File for Full Model 130
B.3.1 Capacitor $C1$ Switching Case 130
B.3.2 Balanced Three-Phase to Ground Fault Case 133
B.4 ATP Data File for Robust TLNE Model 136
B.4.1 Capacitor $C1$ Switching Case 136
B.4.2 Balanced Three-Phase to Ground Fault Case 140

Appendix C AIES Area 50 Backbone **145**
C.1 AIES Area 50 Backbone Diagram in ATP 145
C.2 PSS/E Procedures in Obtaining Equivalents for Area 50 Backbone 148
C.3 Area 50 Backbone ATP Data Files 149

C.3.1 Full Model 149
C.3.2 Robust TLNE Model 153

Appendix D EMTP Models of Passive Elements **157**
D.1 *L* Branch 157
D.2 *C* Branch 158
D.3 *RL* Branch 158
D.4 *RC* Branch 158
D.5 *LC* Branch 158
D.6 *RLC* Branch 159
D.7 *RLCG* Branch 159

Appendix E C++ EMTP S-function Complete Source Code **160**
E.1 `emtp.h` 161
E.2 `emtp.cpp` 177

List of Tables

2.1 Partial derivatives with respect to parameters of $Z_{eq}(s)$ in surface layer . 34

2.2 Partial derivatives with respect to parameters of diagonal elements in deep region admittance matrix 34

2.3 Partial derivatives with respect to parameters of off-diagonal elements in deep region admittance matrix 35

2.4 Example 1 computational time comparison at time-step size 10μs 42

3.1 Example 2 computational time comparison at time step size 20μs 62

4.1 AIES Area 50 computational time comparison at time step size 20μs . . . 84

List of Figures

1.1 Study zone and external system 3
1.2 Two-layer network equivalent for external system 4

2.1 Frequency scan of single-phase single-port network 10
2.2 Frequency scan of single-phase multi-port network 10
2.3 Synthesis of RL and $RLCG$ branches in FDNE 12
2.4 Marti's frequency-dependent line model 15
2.5 RC network realization of $Z_{eq}(\omega)$ approximating $Z_c(\omega)$ 16
2.6 One RC block and its discretization 16
2.7 EMTP equivalent circuit for Marti's frequency-dependent line model . . 18
2.8 Flowchart of Genetic Algorithms 20
2.9 Frequency response of characteristic impedance $Z_c(\omega)$ and its low-order approximation 22
2.10 Frequency response of back-winded weighting function $P(\omega)$ and its low-order approximation 23
2.11 Flowchart of Genetic Algorithm for Robust TLNE 28
2.12 Example 1 system diagram and its partitioning 40
2.13 Example 1 RMS-error% of input admittance versus deep region order . . 40
2.14 Example 1 TL1 characteristic impedance 43
2.15 Example 1 deep region admittance 43
2.16 Example 1 input admittance 44
2.17 Example 1 transient simulation and comparison 45

3.1 Robust TLNE model for external system with active elements 47
3.2 Obtaining Norton equivalent current sources for external system 48

3.3 Flowchart for obtaining Robust TLNE model for generic external systems 50
3.4 Example 2 system. (a) Example 2 system diagram and its partitioning. (b) Example 2 passive part of the external system. 54
3.5 Example 2 aerial mode RMS-error% of input admittance *v.s.* deep region order 55
3.6 Example 2 ground mode input admittance $\mathbf{Y}_{input,0,11}$ 56
3.7 Example 2 ground mode input admittance $\mathbf{Y}_{input,0,12}$ 56
3.8 Example 2 ground mode input admittance $\mathbf{Y}_{input,0,22}$ 57
3.9 Example 2 aerial mode input admittance $\mathbf{Y}_{input,\alpha,11}$ 57
3.10 Example 2 aerial mode input admittance $\mathbf{Y}_{input,\alpha,12}$ 58
3.11 Example 2 aerial mode input admittance $\mathbf{Y}_{input,\alpha,22}$ 58
3.12 Example 2 aerial mode deep region admittance $\mathbf{Y}_{deep,\alpha,11}$ 59
3.13 Example 2 aerial mode deep region admittance $\mathbf{Y}_{deep,\alpha,12}$ 59
3.14 Example 2 aerial mode deep region admittance $\mathbf{Y}_{deep,\alpha,22}$ 60
3.15 Example 2 TL1 aerial mode characteristic impedance 60
3.16 Example 2 TL2 aerial mode characteristic impedance 61
3.17 Example 2 TL3 aerial mode characteristic impedance 61
3.18 Example 2 $C1$ switching transient simulation and comparison (phase A) 64
3.19 Example 2 $C1$ switching transient simulation and comparison (three phase) 65
3.20 Example 2 three-phase to ground fault transient simulation and comparison (phase A) 66
3.21 Example 2 three-phase to ground fault transient simulation and comparison (three phase) 67

4.1 Map of AIES (Courtesy of AESO) 69
4.2 AIES Area 50 Backbone Single-Line Diagram 70
4.3 Building a frequency-dependent transmission line from TASMo database in ATPDraw 74
4.4 A part of AIES Area 50 diagram in ATPDraw 75
4.5 AIES Area 50 aerial mode RMS-error% of input admittance versus deep region order 76
4.6 AIES Area 50 ground mode input admittance $\mathbf{Y}_{input,0}$ 77

4.7 AIES Area 50 aerial mode input admittance $\mathbf{Y}_{input,\alpha}$ 77
4.8 AIES Area 50 aerial mode deep region admittance $\mathbf{Y}_{deep,\alpha,11}$ 78
4.9 AIES Area 50 aerial mode deep region admittance $\mathbf{Y}_{deep,\alpha,12}$ 78
4.10 AIES Area 50 aerial mode deep region admittance $\mathbf{Y}_{deep,\alpha,22}$ 79
4.11 AIES Area 50 1202L aerial mode characteristic impedance 79
4.12 AIES Area 50 1203L aerial mode characteristic impedance 80
4.13 AIES Area 50 1209L aerial mode characteristic impedance 80
4.14 AIES Area 50 three-phase to ground fault transient simulation and comparison (phase *A*) 82
4.15 AIES Area 50 three-phase to ground fault transient simulation and comparison (three phase) 83

5.1 Hardware architecture of the RTX-LAB real-time simulator [33] 87
5.2 Software architecture of the RTX-LAB real-time simulator [33] 88
5.3 C++ class hierarchy for EMTP 90
5.4 C++ data type hierarchy for EMTP 95
5.5 C++ function hierarchy for reading ATP data files 96
5.6 SIMULINK S-function flowchart for EMTP 98
5.7 Example 2 SIMULINK diagram for S-function based EMTP 101
5.8 Example 2 real-time simulation result (×1k) of capacitor switching (phase *A*) 102
5.9 Example 2 real-time simulation voltage result (×1k) of capacitor switching (three-phase) 103
5.10 Example 2 real-time simulation current result (×1k) of capacitor switching (three-phase) 104
5.11 Example 2 real-time simulation result (×1k) of balanced fault (phase *A*) 105
5.12 Example 2 real-time simulation voltage result (×1k) of balanced fault (three-phase) 106
5.13 Example 2 real-time simulation current result (×1k) of balanced fault (three-phase) 107
5.14 AIES Area 50 SIMULINK diagram for S-function based EMTP 109
5.15 AIES Area 50 real-time simulation result (×1k) of balanced fault (phase *A*) 110

5.16 AIES Area 50 real-time simulation voltage result (×1k) of balanced fault (three phase) 111
5.17 AIES Area 50 real-time simulation current result (×1k) of balanced fault (three phase) 112

D.1 Norton equivalent circuit of the discretized branch of passive elements . 157

List of Symbols

A, B, P, ...	Capital boldface letters denote matrices and phaser vectors
u, y, z, ...	Lowercase boldface letters denote vectors
$u, y, z, ...$	Lowercase italics denote scalar-valued function or scalars
$\overrightarrow{V}$	Voltage phasor
$\overrightarrow{I}$	Current phasor
$\overrightarrow{S}$	Complex power
$\mathbf{Y}^T$	Transpose of $\mathbf{Y}$
$\mathbf{Y}^*$	Complex conjugate and transpose of $\mathbf{Y}$
$\mathrm{Re}(\mathbf{Y})$	Real part of $\mathbf{Y}$
$\mathrm{Im}(\mathbf{Y})$	Imaginary part of $\mathbf{Y}$
$\|\mathbf{Y}\|_F$	Frobenius norm of $\mathbf{Y}$
$\mathrm{eig}(\mathbf{Y})$	Eigenvalues of $\mathbf{Y}$
Y or $\mathbf{Y}$	Admittance or admittance matrix
Z or $\mathbf{Z}$	Impedance or Impedance matrix
V or $\mathbf{V}$	Voltage phaser or phaser vector
I or $\mathbf{I}$	Current phaser or phaser vector
$v(t)$ or $\mathbf{v}(t)$	Instantaneous voltage or voltage vector
$i(t)$ or $\mathbf{i}(t)$	Instantaneous current or current vector
$\mathbf{Y}_{input}$	Original external system input admittance matrix
$\tilde{\mathbf{Y}}_{input}$	Admittance matrix approximating external system
$\tilde{\mathbf{Y}}^0_{input}$	First approximation of external system input admittance matrix
$\mathbf{Y}_{surface}$	Original surface layer admittance matrix
$\tilde{\mathbf{Y}}_{surface}$	Admittance matrix approximating surface layer

$\tilde{\mathbf{Y}}^{0}_{surface}$	First approximation of surface layer admittance matrix
$\mathbf{Y}_{deep}$	Original deep region admittance matrix
$\tilde{\mathbf{Y}}^{VF}_{deep}$	Original deep region admittance matrix fitted by Vector Fitting
$\tilde{\mathbf{Y}}_{deep}$	Admittance matrix approximating deep region
$\tilde{\mathbf{Y}}^{0}_{deep}$	First approximation of deep region admittance matrix

List of Acronyms

EMTP	Electro-Magnetic Transients Program
ATP	Alternative Transients Program
FDNE	Frequency-Dependent Network Equivalent
TLNE	Two-Layer Network Equivalent
GA	Genetic Algorithm
VF	Vector Fitting
RMS	Root-Mean-Square
SISO	Single-Input-Single-Output
MIMO	Multiple-Input-Multiple-Output
QP	Quadratic Programming
SQP	Sequential Quadratic Programming
NLP	Nonlinear Programming
CLLSQ	Constrained Linearized Least-Square
LTI	Linear and Time-Invariant
AIES	The Alberta Interconnected Electric System
TASMo	Transmission Administrator System Model
RTW	Real-Time Workshop
RTOS	Real-Time Operating System
FPGA	Field-Programmable Gate Array

1 Introduction

Electromagnetic transients in power systems, induced by switchings, surges, faults and other topology changes in the network, occur in a very short period of time. These transients can activate control and protection systems, lead to power interruptions, or even result in component failures. Studies of such transients are thus of great importance.

Via time-domain, the simulation can be carried out in off-line or real-time. The off-line digital simulations by Electro-Magnetic Transients Program (EMTP) [1, 2], *e.g.*, ATP, PSCAD/EMTDC, EMTP-RV, MICROTRAN, NETOMAC, *etc.*, are able to verify the design of line and station insulation and selection of equipment. Nonetheless, real-time simulation is required in order to realize

- Accurate design and testing of new apparatus and schemes such as controllers, relays, and other protective equipment like surge arrestor, spark gaps, *etc.*,
- Closed loop behavior study of the device on the full system,
- Studying the high frequency phenomenon due to the introduction of different disturbance and switching actions, and
- Training and education purposes.

A variety of analogue and digital real-time simulators have so far been developed and

used at research institutes, universities, power companies and manufacturers all over the world, such as Micro-Network of EDF (France), AC-DC power system simulator at CRIEPI (Japan), Hydro-Québec real-time simulator (Canada), HVDC simulator of CEPEL (Brazil), APSA of Kansai Electric Power Company (Japan), RTDS of Manitoba HVDC Research Center (Canada), *etc.*. However, the real-time digital simulation of large power systems are limited to only analogue or hybrid (mixed digital and analogue) simulators. This is due to the excessive computational time during the simulation of the full representation of large power system. In such a case, even detailed off-line simulation is computationally prohibitive, since the system model is too complicated. Simulation of transients for large power systems, especially in real-time, requires not only blazing computational power but also simpler models. In order to find more computational efficient representation of power systems, considerable efforts have been made by the past three decades. The main achievement is study zone *v.s.* external system, Frequency-Dependent Network Equivalents (FDNE), and Two-Layer Network Equivalents (TLNE).

1.1 Frequency-Dependent Network Equivalent

In electromagnetic transient studies, due to system complexities, it is a common practice that the system is divided into a *study zone*, a restricted part of the system where the transient phenomena occur and whose components must be fully characterized including any nonlinear and time-variant elements, and an *external system* which encompasses the rest of the system. External system is considered to be electrically remote from the transient location. Higher frequency electromagnetic waves, due to higher attenuation, propagate shorter electrical distances. Therefore, the external system is represented by a linear equivalent network, *i.e.*, Frequency-Dependent Network Equivalent (FDNE), which is shown in Fig. 1.1. There is no well-defined approach for the division of study zones and external systems. Instead, engineering judgement plays an important role. CIGRÉ provides guidelines in [4].

There have been quite a few successful achievements in constructing FDNE which greatly reduce the computational burden of transient simulations [5–20]. Those FDNE models can mainly be classified as frequency-domain model relying on convolutions in

time-domain simulation [5], z-domain models [6–9] and s-domain models [10–22] for EMTP. The latest development in s-domain fitting is Vector Fitting (VF) by Gustavsen *et al.* [17–20]. Time-domain fitting for Sparse Network Equivalent (SNE) due to Boaventura *et al.* [8,9] is the up-to-date fitting method in z-domain.

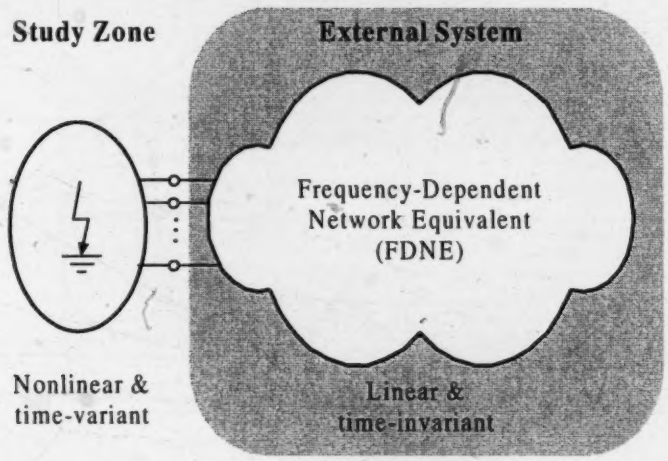


Figure 1.1: Study zone and external system

1.2 Two-Layer Network Equivalent

In real-time simulation of large power systems, however, the bottleneck still remains due to the efficiency concern in convolutions and the high order of the fitted FDNE mathematical model of external systems. It is obvious that an alternative way must be found out to reduce the model complexity without significant effect on accuracy. Proposed by Abdel-Rahman *et al.* [21,22], the Two-Layer Network Equivalent (TLNE) model is another milestone in overcoming the obstacle of real-time digital simulations. It has been noticed that in frequency domain, the fact that external system has the property of numerous resonance peaks is primarily due to frequency-dependent transmission lines. If the leading part of the external system is retained as reduced-order line models, it is possible to find an FDNE model to compensate the deviations due to the order reduction of the line models and the rest of the external system. Thus, illustrated in Fig. 1.2, we have a TLNE model consisting of a *surface layer* represented by reduced-order frequency-dependent transmission line models and a *deep region* as a low-order

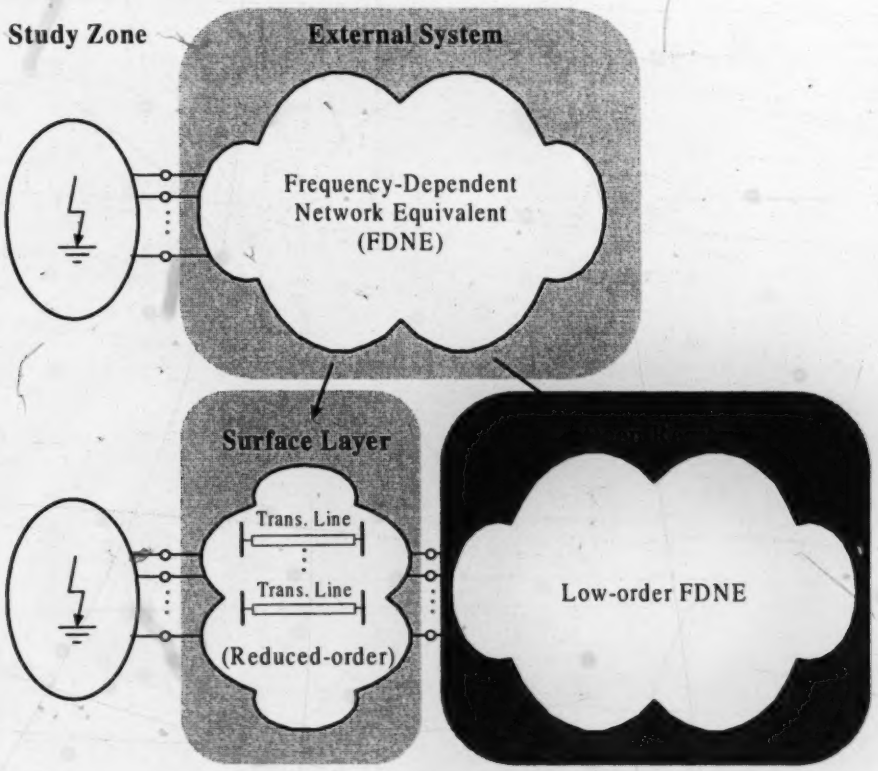


Figure 1.2: Two-layer network equivalent for external system

FDNE. The impact of surface layer and deep region on the external system input admittance varies with frequency. At low frequencies, both surface layer and deep region contributes to the admittance. When the frequency increases, however, the contribution of deep region diminishes drastically due to high attenuation. Therefore, the FDNE model for deep region is able to be obtained in low order. The boundary between surface layer and deep region also heavily relies on engineering judgement and experience. Some guidelines should be obeyed to obtain better and simpler possible equivalent [21]:

- The topology of surface layer, which includes loads and other shunt connections, is retained. This is because removing such elements in surface layer will cause major deviations from the original system in frequency response, which is very unlikely to be compensated by the developed model with stability and passivity constraints.

- Deep region is obtained based on attenuation in the transmission lines of surface layer. Such damping effects depend on the length of transmission lines and the loads in surface layer. Therefore, if lines are short and surface layer does not contain loads or other elements that produce damping, the surface layer is considered to include more lines.

- If difficulties arises in finding deep region parameters, more buses should be included in surface layer, *i.e.*, the surface layer is made deeper.

In addition,

- Depending on the size and complexities, big and complicated external systems result in complex frequency responses. Therefore more lines should be included in surface layer (surface layer is deeper). For example, in modeling a large power system, only one transmission line in surface layer will result in a high-order deep region model, which cancels off the benefits of TLNE model.

1.3 Existing Issues Associated with TLNE

In obtaining TLNE by existing approach [21, 22], it is however revealed following concerns:

- With low-order VF in obtaining the deep region, it is difficult to control deviations with respect to the original deep region, which Sequential Quadratic Programming (SQP) may not be able to compensate.

- Frequency response at DC is not specifically accentuated although it affects transient DC offset.

- SQP is prone to divergence. If better first approximations of input admittance of external systems can be found, SQP can be replaced by Constrained Linearized Least-Square (CLLSQ) optimization, whose the convergence is guaranteed.

- Optimization of surface layer is especially helpful in increasing accuracy of frequency response in the low frequency range, *e.g.*, DC and power frequency, with little cost of computational time. Thus, parameters of surface layer are optimized.

- In multi-port external systems with complex frequency response, the passivity constraint is very strong, so the freedom for changing the parameters is small. Therefore, the first approximation in a multi-port case, is required to be closer to the original than that for single-port case. Thus, transmission line parameters in surface layer require higher accuracy but low-order realization methods.
- The order of deep region is not determined in an optimal way. Optimal deep region order can be obtained by comparing a series orders of deep region with each other.

1.4 Thesis Objectives

Considering the requirement of developing appropriate an external system model for real-time digital simulations and possible improvements in the TLNE model, the main objectives of this thesis are:

- To extend the concept of TLNE model, *i.e.*, Robust TLNE model, which is robust in terms of stability and passivity, more accurate and computationally efficient.
- To compare the performance of the Robust TLNE model with that of the full EMTP model and the FDNE model based on case studies.
- To obtain an TLNE model for the Alberta Interconnected Electric System (AIES) which is suitable for real-time digital simulation of electromagnetic transients.
- To implement a customized EMTP program in C++ for real-time simulation purpose.
- To realize real-time digital simulation of large power systems such as AIES.

1.5 Thesis Outline

The rest of this thesis consists of the following chapters:

- Chapter 2 formulates the Robust TLNE model for passive networks. The background concepts and methods such as passivity criterion, frequency scan, Vector

Fitting (VF), Marti's frequency-dependent line model, and Genetic Algorithms (GAs) are covered as well. One simple system is examined.

- Chapter 3 extends the Robust TLNE model to three-phase active networks. A modified benchmark system which is both multi-port and multi-phase verifies the approach. Performance analysis is carried out in terms of accuracy and computational efficiency.
- Chapter 4 constructs the full model of AIES Area 50 (Backbone) and its robust TLNE model in ATP. Procedures and methods in obtaining the models are explained in detail. Computational performance and accuracy are also analyzed.
- Chapter 5 presents the real-time digital simulation of AIES at Bus 524 Genesee by a Xeon-cluster based real-time simulator at RTX-LAB, which is implemented by a MATLAB/SIMULINK C++ S-function program.
- The conclusions of the thesis and future work are summarized in Chapter 6.

2

Robust Two-Layer Network Equivalent for Passive Networks

In this chapter, detailed theory and formulation of existing TLNE model and the Robust TLNE model are explained. Divided into a surface layer and a deep region, an external system is further reduced to a Robust TLNE with high accuracy and efficiency. The most significant improvements include the implementation of Genetic Algorithms (GAs), Constrained Linearized Least-Square (CLLSQ) optimization and optimal deep region order determination [23].

2.1 Models and Concepts in the Robust TLNE

Before going further into the model, the following topics are briefly explained as the TLNE model is founded on the following methods, concepts and models: passivity criterion, frequency scan, Vector Fitting (VF) used in obtaining FDNE, Marti's frequency-dependent transmission line model and Genetic Algorithms (GAs).

2.1.1 Positive-Realness and Passivity Criterion

An electrical network is passive if it consumes real power for any active sources applied to input terminals, and does not deliver real power. Passivity affects the stability

of time-domain simulation. An obvious example is a linear network without any active sources. In realistic networks, passivity is not of concern. However, in fitting the frequency response of passive networks, especially large networks, passivity is of great importance. The fitted model representing such network is required to be passive as well. The electrical network of FDNE model with passivity violation will more likely result in unstable and erroneous simulations.

For an electrical network whose admittance matrix is $\mathbf{Y}$,

$$\mathbf{YV} = \mathbf{I} \tag{2.1}$$

defines the relationship between current vector $\mathbf{I}$ applied to network terminals and voltage vector $\mathbf{V}$ corresponding to each terminal. The real power absorbed by the network is

$$P = \mathrm{Re}(\mathbf{V}^*\mathbf{I}) = \mathrm{Re}(\mathbf{V}^*\mathbf{YV}) = \mathrm{Re}(\mathbf{V}^*(\mathbf{G} + j\mathbf{B})\mathbf{V}) = \mathrm{Re}(\mathbf{V}^*\mathbf{GV}) \tag{2.2}$$

where the asterisk * stands for transpose and conjugate. Notice that the final result of (2.2) is a quadratic form. Therefore, the real power is absorbed or $P > 0$ if and only if the conductance matrix $\mathbf{G} = \mathrm{Re}\{\mathbf{Y}\}$ is positive-definite. It follows that since $\mathbf{G}$ matrix representing a linear network is real and symmetric, all eigenvalues of $\mathbf{G}$ must be real. Thus, passivity criterion can be equivalenced to requiring all eigenvalues of $\mathbf{G}$ to be positive or

$$\mathrm{eig}(\mathbf{G}) > \mathbf{0}. \tag{2.3}$$

The passivity criterion of electrical networks is also denoted as *positive-real* criterion in linear system theory.

2.1.2 Frequency Scan of Linear Passive Networks

To obtain the frequency response of a linear network, frequency scan is the appropriate method. In single-phase single-port network case, a current source with unity magnitude and zero phase angle is applied to the designated port, which is shown in Fig. 2.1. The node voltage of the port is measured. At a particular frequency, based on the phasor equation $Z = V/I$, since the current has unity magnitude and zero phase shift, the voltage phasor measured is equal to impedance Z. Thus frequency response of impedance $Z(\omega)$ is obtained by measuring all voltage phasors when applying the current source

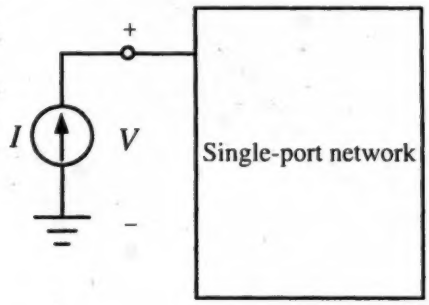


Figure 2.1: Frequency scan of single-phase single-port network

with a range of frequencies of interest. The admittance $Y(\omega)$ is found by taking the inverse of $Z(\omega)$ at each frequency point.

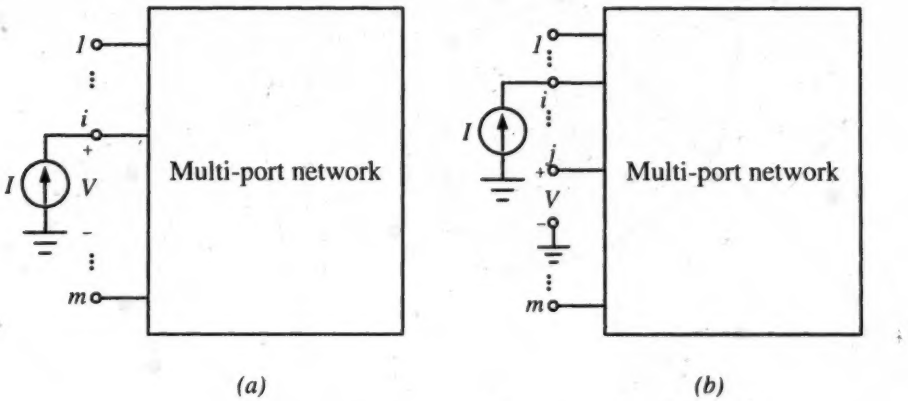


Figure 2.2: Frequency scan of single-phase multi-port network

When frequency scan is applied to a multi-port network, the situation is a little bit different. For a m-port network, we are looking for an impedance matrix that has following form

$$\mathbf{Z}(\omega) = \begin{bmatrix} Z_{11}(\omega) & \cdots & Z_{1m}(\omega) \\ \vdots & \ddots & \vdots \\ Z_{m1}(\omega) & \cdots & Z_{mm}(\omega) \end{bmatrix} \tag{2.4}$$

Since the system is linear, $\mathbf{Z}(\omega)$ is symmetric. Thus off-diagonal elements $Z_{ij}(\omega) = Z_{ji}(\omega)$ where $1 < (i, j) < m, i \neq j$. For diagonal elements $Z_{ii}(\omega)$ where $1 < i < m$, they can be obtained by applying a current with unity magnitude and zero phase angle at i-th port while leaving all other ports as open circuit and measuring the voltage phasor

at i-th port, as shown in Fig. 2.2(a). The same method is applied to the off-diagonal element $Z_{ij}(\omega)$ where $1 < (i,j) < m$, $i \neq j$, except the voltage phasor is measured at j-th port instead of i-th port. Finally, admittance matrix $\mathbf{Y}(\omega)$ is obtained by inverting $\mathbf{Z}(\omega)$ matrix in (2.4) at each frequency point, which is

$$\mathbf{Y}(\omega) = \begin{bmatrix} Y_{11}(\omega) & \cdots & Y_{1m}(\omega) \\ \vdots & \ddots & \vdots \\ Y_{m1}(\omega) & \cdots & Y_{mm}(\omega) \end{bmatrix} \tag{2.5}$$

where $Y_{ij}(\omega) = Y_{ji}(\omega)$.

2.1.3 Vector Fitting

Provided Linear and Time-Invariant (LTI), an external system can be reproduced by a rational transfer function in frequency domain for the Single-Input-Single-Output (SISO) case, or in the Multiple-Input-Multiple-Output (MIMO) case, by a rational transfer function matrix whose elements share a common denominator, *i.e.*, same poles. Such procedures require curve fitting in frequency domain. Vector Fitting (VF) [17–20] is the appropriate technique used in s-domain. Since expressing the rational transfer function (matrix) in partial fraction form is suitable for order reduction as well as its robustness and effectiveness, VF is applied in FDNE and TLNE model [21, 22]. In MIMO case, the fitted rational matrix having a common set of poles is very useful not only in improving the computational efficiency, but also in maintaining the simplicities of the realistic electrical network branches in EMTP. The method is able to fit the frequency responses with very high accuracy on the RMS-error% basis. For an m-port network system, the fitted proper (the order of denominator is equal to that of numerator) n-th order rational function matrix with common poles is expressed in partial fraction form as

$$\mathbf{Y}_{VF} = \begin{bmatrix} d_{11} + \sum\limits_{i_{11},j=1}^{n} \frac{c_{i_{11}}}{s-a_j} & d_{12} + \sum\limits_{i_{12},j=1}^{n} \frac{c_{i_{12}}}{s-a_j} & \cdots & d_{1m} + \sum\limits_{i_{1m},j=1}^{n} \frac{c_{i_{1m}}}{s-a_j} \\ d_{21} + \sum\limits_{i_{21},j=1}^{n} \frac{c_{i_{21}}}{s-a_j} & d_{22} + \sum\limits_{i_{22},j=1}^{n} \frac{c_{i_{22}}}{s-a_j} & \cdots & d_{2m} + \sum\limits_{i_{2m},j=1}^{n} \frac{c_{i_{2m}}}{s-a_j} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} + \sum\limits_{i_{m1},j=1}^{n} \frac{c_{i_{m1}}}{s-a_j} & d_{m2} + \sum\limits_{i_{m2},j=1}^{n} \frac{c_{i_{m2}}}{s-a_j} & \cdots & d_{mm} + \sum\limits_{i_{mm},j=1}^{n} \frac{c_{i_{mm}}}{s-a_j} \end{bmatrix} \tag{2.6}$$

where d_{ij} $(1 \leq i,j \leq m)$ is constant term; $c_{i_{jk}}$ $(1 \leq i \leq n,\ 1 \leq j,k \leq m)$ and a_i $(1 \leq i \leq n)$ are residue and pole, respectively. Notice that the poles and residues may come in

complex conjugate pairs.

2.1.4 Electrical Network Realization

The fitted admittance matrix $\mathbf{Y}_{VF}$ in (2.6) can be converted to a realistic electrical network, which has branches between nodes and between nodes and the ground. Branch admittance between node i and ground is given as [19]

$$Y_i(s) = \sum_{j=1}^{m} (\mathbf{Y}_{VF})_{ij} \tag{2.7}$$

and branch admittance between node i and node j is

$$Y_{ij}(s) = -(\mathbf{Y}_{VF})_{ij} \,. \tag{2.8}$$

Since all elements of $\mathbf{Y}_{VF}$ share the same set of poles, the summation in (2.7) becomes the sum of d terms as well as the sum of c terms in the corresponding poles in (2.6).

Each branch calculated in (2.7) and (2.8) gives a rational function

$$Y(s) = d + \sum_{i=1}^{n} \frac{c_i}{s - a_j}. \tag{2.9}$$

Electrical realization generates RL and $RLCG$ branches [10, 11, 19], shown in Fig. 2.3. The R_0 is calculated as

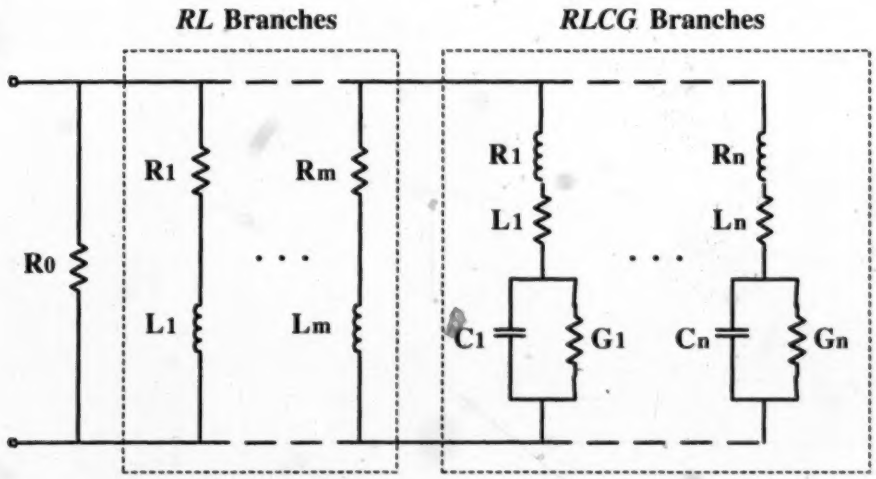


Figure 2.3: Synthesis of RL and $RLCG$ branches in FDNE

$$R_0 = 1/d \tag{2.10}$$

and each $\frac{c}{s-a}$ of real pole gives an RL branch

$$R = -a/c,\ L = 1/c \tag{2.11}$$

whereas each complex conjugate pair $\frac{c'+jc''}{s-(a'+ja'')} \frac{c'-jc''}{s-(a'-ja'')}$ gives an $RLCG$ branch

$$\begin{aligned} R &= [-2a' + 2(c'a' + c''a'')L]\,L \\ L &= 1/(2c') \\ C &= \frac{1}{[(a')^2 + (a'')^2 + 2(c'a' + c''a'')R]\,L} \\ G &= -2(c'a' + c''a'')CL \end{aligned} \tag{2.12}$$

The R, L, C, G in (2.10), (2.11), (2.12) may appear as negative values. However, EMTP packages such as ATP and PSCAD/EMTDC accept those values. As long as the passivity of overall FDNE is guaranteed, correct and stable simulation is ensured.

2.1.5 Frequency-Dependent Transmission Line Model

In transient simulation of power system networks, accurate modeling requires both frequency dependent factors due to skin effect and traveling wave effects in transmission lines be taken into account. The constant-parameter line model [1] tends to exaggerate the transients somehow. Many efforts have been devoted in the past three decades to the development of frequency-dependent line models in EMTP [24–28] and simulations of transmission line transients. The line models can be categorized as modal-domain models [24, 25] and phase-domain models [26–29]. Since there have been successful approaches in the real-time application of the low-order model in [38–41], Marti's line model [25] is employed in the Robust TLNE model.

When the frequency dependence of parameters and the distributed nature of losses in transmission lines are taken into account, it is very difficult to write the line equations directly in time domain. However, the solution can be readily obtained in frequency domain by the well-known equations:

$$V_k(\omega) = \cosh[\gamma(\omega)\ell]V_m(\omega) - Z_c(\omega)\sinh[\gamma(\omega)\ell]I_m(\omega) \tag{2.13a}$$

$$I_k(\omega) = \frac{1}{Z_c(\omega)}\sinh[\gamma(\omega)\ell]V_m(\omega) - \cosh[\gamma(\omega)\ell]I_m(\omega) \tag{2.13b}$$

where V_k, V_m, I_k and I_m are the frequency-domain quantities corresponding to sending-end and receiving-end voltages and currents, respectively; ℓ is the line length; $Z_c(\omega)$

and $\gamma(\omega)$ are frequency-dependent characteristic impedance and propagation function defined as

$$Z_c(\omega) = \sqrt{\frac{R(\omega) + j\omega L(\omega)}{G + j\omega C}};\ \gamma(\omega) = \sqrt{(R(\omega) + j\omega L(\omega))(G + j\omega C)} \tag{2.14}$$

with

$R(\omega)$	=	series resistance	$L(\omega)$	=	series inductance
G	=	shunt conductance	C	=	shunt capacitance.

To relate currents and voltages in frequency domain, new functions are defined [25]: Forward traveling functions

$$F_k(\omega) = V_k(\omega) + Z_c(\omega)I_k(\omega) \tag{2.15a}$$

$$F_m(\omega) = V_m(\omega) + Z_c(\omega)I_m(\omega) \tag{2.15b}$$

and backward traveling functions

$$B_k(\omega) = V_k(\omega) - Z_c(\omega)I_k(\omega) \tag{2.16a}$$

$$B_m(\omega) = V_m(\omega) - Z_c(\omega)I_m(\omega). \tag{2.16b}$$

By eliminating $V_k(\omega)$, $V_m(\omega)$, $I_k(\omega)$ and $I_m(\omega)$ from (2.13a), (2.13b), (2.16a) and (2.16b), we obtain

$$B_k(\omega) = A_1(\omega)F_m(\omega) \tag{2.17a}$$

$$B_m(\omega) = A_1(\omega)F_k(\omega) \tag{2.17b}$$

where

$$A_1(\omega) = e^{-\gamma(\omega)\ell} = \frac{1}{\cosh[\gamma(\omega)\ell] + \sinh[\gamma(\omega)\ell]} \tag{2.18}$$

the time-domain form of which is defined as the *weighting function* $a_1(t)$, obtained by the inverse Fourier transform of $A_1(\omega)$. Equations (2.16a) and (2.16b) gives the Thevenin equivalent network shown in Fig. 2.4. The voltage sources $b_k(t)$ and $b_m(t)$ are the time domain forms of (2.17a) and (2.17b), which are convolution integrals

$$b_k(t) = \int_\tau^\infty f_m(t-u)a_1(u)du \tag{2.19a}$$

$$b_m(t) = \int_\tau^\infty f_k(t-u)a_1(u)du \tag{2.19b}$$

where

$$f_k(t) = 2V_k(t) - b_k(t) \tag{2.20a}$$

$$f_m(t) = 2V_m(t) - b_m(t). \tag{2.20b}$$

The reason that the lower limit of those integrals is the propagation delay τ is that in time domain an impulse on one end of the line will not reach the other end until the time of τ [25].

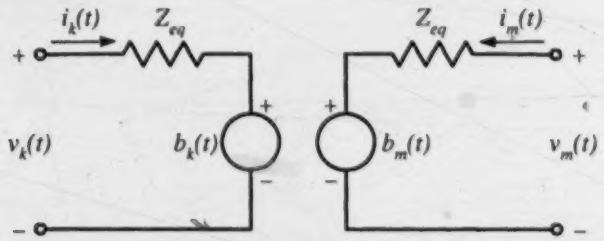


Figure 2.4: Marti's frequency-dependent line model

Accuracy of Marti's line model greatly depends on the fitting of $Z_c(\omega)$ and $A_1(\omega)$. The appropriate techniques used are Bode's asymptotic fitting technique [25] and non-linear Levenberg-Marquardt (LM) fitting method due to Fernandes *et al.* [42,43]. $Z_c(\omega)$ is fitted by an n-th order rational function of the form

$$Z_{eq}(s) = k_0 \frac{(s+z_1)(s+z_2)\cdots(s+z_n)}{(s+p_1)(s+p_2)\cdots(s+p_n)} \tag{2.21}$$

with all poles and zeros are real and simple and lie on the left hand side of the complex plane (a minimal phase system) and is further expanded to partial fraction form

$$Z_{eq}(s) = k_0 + \sum_{i=1}^{n} \frac{r_i}{s+p_i} \tag{2.22}$$

which is realized by a series of RC parallel blocks (Foster I network realization, Fig. 2.5). The parameters are calculated as

$$R_0 = k_0,\ R_i = k_i/p_i,\ C_i = 1/k_i. \tag{2.23}$$

One RC parallel block shown in Fig. 2.6(a) can be discretized into an equivalent resistor in series with a voltage source (Fig. 2.6(b)) [1]. The discrete nodal equation is

$$v(t) = R_{RC}i(t) + V(t-\Delta t) \tag{2.24}$$

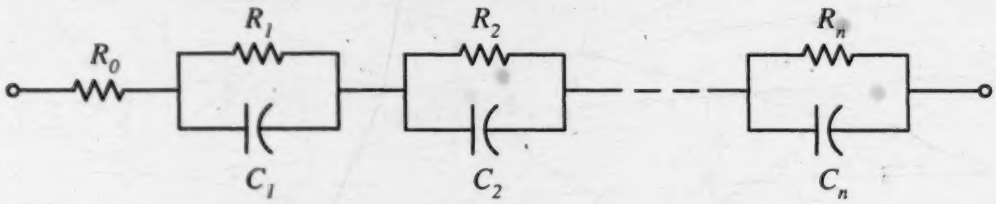


Figure 2.5: RC network realization of $Z_{eq}(\omega)$ approximating $Z_c(\omega)$

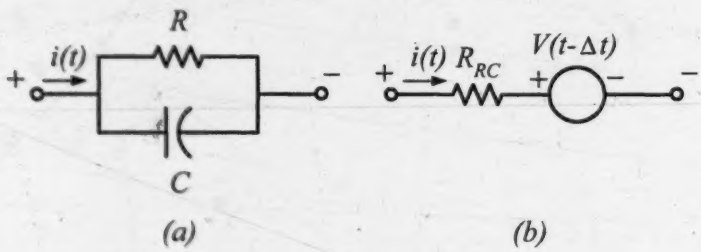


Figure 2.6: One RC block and its discretization

where

$$R_{RC} = \frac{R\frac{\Delta t}{2C}}{R + \frac{\Delta t}{2C}} \tag{2.25}$$

$$V(t-\Delta t) = \frac{R^2\frac{\Delta t}{C}}{(R+\frac{\Delta t}{2C})^2} i(t-\Delta t) + \frac{R-\frac{\Delta t}{2C}}{R+\frac{\Delta t}{2C}} V(t-2\Delta t). \tag{2.26}$$

Since the past history term $V(t-\Delta t)$ is only related to the current $i(t-\Delta t)$ and history term $V(t-2\Delta t)$ of the previous time step, the RC parallel block networks representing Z_{eq} in Fig. 2.5 can be further reduced into an equivalent resistor

$$R_{eq} = R_0 + \sum_{i=1}^{n} \frac{R_i\frac{\Delta t}{2C_i}}{R_i + \frac{\Delta t}{2C_i}} \tag{2.27}$$

in series with a voltage source

$$V_{eq}(t-\Delta t) = \sum_{i=1}^{n} V_i(t-\Delta t) \tag{2.28}$$

where

$$V_i(t-\Delta t) = \frac{R_i^2\frac{\Delta t}{C_i}}{(R_i+\frac{\Delta t}{2C_i})^2} i_i(t-\Delta t) + \frac{R_i-\frac{\Delta t}{2C_i}}{R_i+\frac{\Delta t}{2C_i}} V_i(t-2\Delta t) \tag{2.29}$$

and $1 \le i \le n$, n is total number of RC parallel blocks or the order of the approximation of $Z_c(\omega)$.

Computational efficiency of the convolutions of (2.19a) and (2.19b) may be greatly increased if the weighting function $a_1(t)$ has the form of sum of exponential terms [24]. To do so, the same fitting techniques for $Z_c(\omega)$ are applied in the approximation of weighting function $A_1(\omega)$. However, rather than a proper form of $Z_{eq}(s)$, the $A_1(\omega)$ is first back-winded by the propagation delay τ to produce

$$P(\omega) = A_1(\omega)e^{j\omega\tau} \tag{2.30}$$

and then approximated in a strictly proper manner by a m-th order rational function

$$P_a(s) = k\frac{(s+z_1)(s+z_2)\cdots(s+z_q)}{(s+p_1)(s+p_2)\cdots(s+p_m)} \quad (q<m) \tag{2.31}$$

Partial fraction expansion gives

$$P_a(s) = \sum_{i=1}^{m}\frac{r_i}{s+p_i}. \tag{2.32}$$

Thus we obtain a sum of exponentials from the inverse Fourier transformation as

$$a_{1a}(t) = u(t-\tau)\sum_{i=1}^{m}k_i e^{-p_i(t-\tau)} \tag{2.33}$$

where $u(t-\tau)$ is step response with τ delay.

In order to obtain the current sources $b_k(t)$ and $b_m(t)$ in Fig. 2.4, it is necessary to evaluate convolution integrals from (2.19a) and (2.19b). To reduce the computational burden, the weighting function $a_1(t)$ is expressed as sum of exponential terms. For one generic term $ke^{-p(t-\tau)}u(t-\tau)$ in approximated function $a_{1a}(t)$ of (2.33), the convolution integral is

$$s(t) = \int_{\tau}^{\infty} f(t-u)ke^{-p(u-\tau)}du. \tag{2.34}$$

$s(t)$ in (2.34) can be directly obtained from recursive convolution [24] by

$$s(t) = M\cdot s(t-\Delta t) + P\cdot f(t-\tau) + Q\cdot f(t-\tau-\Delta t) \tag{2.35}$$

where

$$M = e^{-p\Delta t} = \alpha \tag{2.36a}$$

$$P = \frac{k}{p}(1-\frac{1-\alpha}{p\Delta t}) \tag{2.36b}$$

$$Q = \frac{k}{p}(\frac{1-\alpha}{p\Delta t}-\alpha). \tag{2.36c}$$

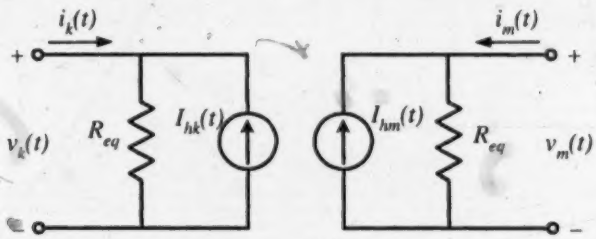


Figure 2.7: EMTP equivalent circuit for Marti's frequency-dependent line model

Consequently, combining (2.28) (2.29), $b_k(t)$ and $b_m(t)$, an EMTP equivalent circuit is obtained, shown in Fig. 2.7. History terms $I_h(t)$ (as both $I_{hk}(t)$ and $I_{hm}(t)$ have the same form of equations) is

$$I_h(t) = \left(V_{eq}(t-\Delta t) + \sum_{j=1}^{m} s_j(t) \right) / R_{eq} \tag{2.37}$$

$$= \left(\sum_{i=1}^{n} V_i(t-\Delta t) + \sum_{j=1}^{m} s_j(t) \right) / R_{eq} \tag{2.38}$$

where

$$V_i(t-\Delta t) = \frac{R_i^2 \frac{\Delta t}{C_i}}{(R_i + \frac{\Delta t}{2C_i})^2} i_i(t-\Delta t) + \frac{R_i - \frac{\Delta t}{2C_i}}{R_i + \frac{\Delta t}{2C_i}} V_i(t-2\Delta t) \tag{2.39}$$

$$s_j(t) = M \cdot s_j(t-\Delta t) + P \cdot f(t-\tau) + Q \cdot f(t-\tau-\Delta t). \tag{2.40}$$

2.1.6 Genetic Algorithms

The Genetic Algorithm (GA) is a probabilistic-rule based global search method mimicking natural biological evolution [48]. As generation-based algorithms, GAs operate on a population of potential solutions, or individuals. By applying the principle of survival of the fittest individuals, GAs are to produce better and better results to a solution in the following generations.

Borrowed from natural genetics by GAs, individuals are encoded as strings, or *chromosomes*, so that unique mappings are created between the *chromosome values* (*genotypes*) and *decision variables* (*phenotype*) of the problem. Chromosomes can be represented by binary, ternary, integer, real value, *etc.* [47, 48]. In order to access the performance, or in genetic terminology — fitness of the individuals, phenotypes obtained by decoding

chromosomes are evaluated in the problem domain by objective functions. The objective functions are used to characterize the individual's performance from decision variables. They establish the basis for selection of pairs of individuals that will be mated together during reproduction [47, 48]. This can be explained as an individual's ability to survive in natural world.

Depending on the problem size and characteristics, a set of individuals or a population is first generated and evaluated by objective functions and each individual is given a fitness value. At each generation, selection phase and reproduction phase are carried out. Based on the fitness values of individuals, the selection biases towards more fit individuals. Relatively more highly fit individuals have a higher probability of being chosen for mating or recombination. The most commonly used methods in selection are Roulette Wheel Selection (RWS) and Stochastic Universal Sampling (SUS) selection [47, 48].

Reproduction phase commonly includes two steps — recombination and mutation. In the recombination step, or crossover step, operators are applied to exchange genetic information between pairs or groups of individuals. Such operations are not necessarily performed on all chromosomes of the population. Instead, a probability is applied. Depending on the encodings of chromosomes, available recombination operation methods are single-point crossover, multi-point crossover, uniform crossover, intermediate recombination, line recombination, *etc.* [48]. Mutation, which is used to converge GAs to global optimum, is applied to new chromosomes generated after recombination. It causes individual chromosomes to change according to probabilistic rules.

Evaluations of the decoded chromosomes of individuals by objective functions are performed to assign fitness values to each individual, so that the selection process for the next generation is to be carried out. After generations of GA procedures, the average performance of individuals of a population is expected to increase, since the less fit individuals are superseded, as what happens in the natural world. Flowchart in Fig. 2.8 illustrates generic procedures.

GAs finish when certain termination criteria are met, *e.g.*, a certain number of generations, or the discovery of the expected chromosomes satisfying the problem [47, 48].

The above discussion shows that GAs has significant difference from the traditional analytical search and optimization methods. They have the merits of [47, 48]

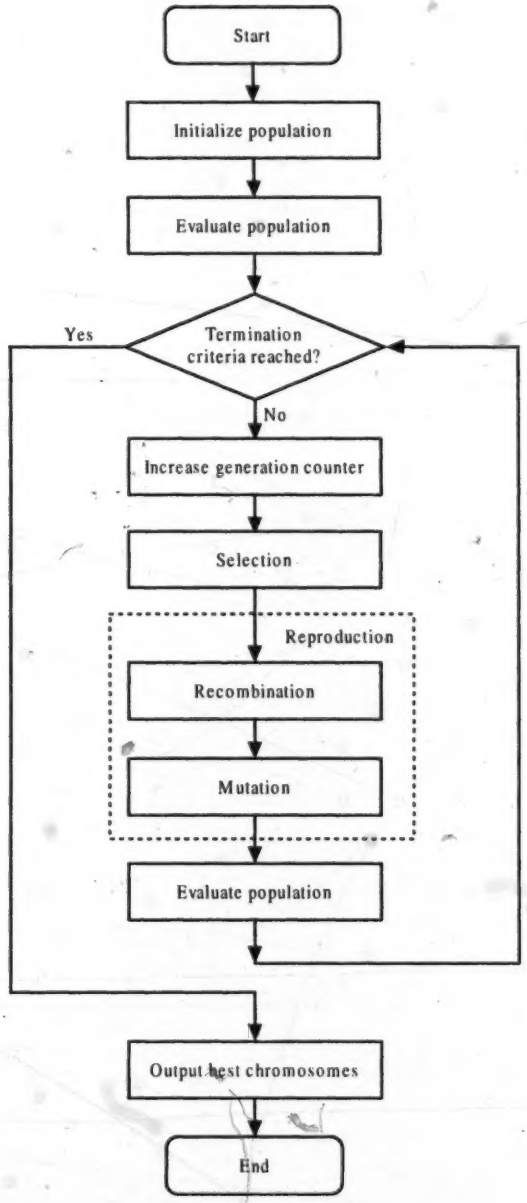


Figure 2.8: Flowchart of Genetic Algorithms

- Search of a population of points instead of single-point search.
- Utilization of objective functions rather than derivative information or other aux-

iliary knowledge.

- Probabilistic rules instead of deterministic ones.
- Encoding the parameters rather than using the parameters themselves.

In the multi-objective optimization problems, GAs are potentially useful for identifying alternative solutions simultaneously. This is particularly helpful in finding best suitable deep region in multi-port external systems. Moreover, its globalism in searching is also important advantage to be utilized from GAs.

2.2 Surface Layer

The surface layer comprises reduced-order or simplified Marti's frequency-dependent line model, which was first proposed by L. Marti [37] used for the simulation of secondary and less important transmission lines. The reduced-order model implies reduced simulation time. Later successful approaches in real-time simulation [38–41] show that since it keeps much of the accuracy with respect to the full-order model, the low-order model is also suitable for real-time implementation.

The order of Marti's line model comprises the order of $Z_{eq}(s)$ from (2.21) approximating characteristic impedance $Z_c(\omega)$ and that of $P_a(s)$ from (2.31) approximating back-winded weighting function $P(\omega)$. For a typical 280km single-phase overhead transmission line of one two-bundle Drake conductor, the frequency response of $Z_{eq}(s)$ and $P_a(s)$ is shown in Figures 2.9 and 2.10. To fit such frequency response by rational functions, Bode's asymptotic technique is employed. Featuring automatic order determination, the ATP Line Constant Program generates an 18th order $Z_{eq}(s)$ and a 16th order $P_a(s)$ for the transmission line within a very low RMS-error% (below 0.3% by default). The generated output data file is listed in Appendix A.3 and the data file format is interpreted in [44].

In this case, the low-order approximation of $Z_c(\omega)$ and that of $P(\omega)$ are also done by applying the same asymptotic technique. The difference is lower order and relatively larger RMS-error% tolerance. The fitting results from ATP listed in Appendix A.4 are 2nd order of $Z_{eq}(s)$ and 1st order of $P_a(s)$, also shown in Figures 2.9 and 2.10. It is obvious that low-order model still maintains much of accuracy. For the purpose of

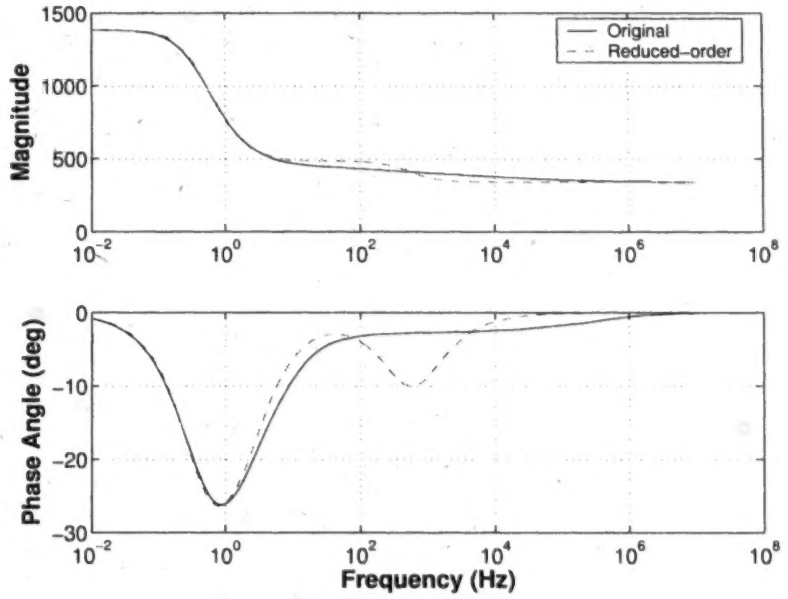


Figure 2.9: Frequency response of characteristic impedance $Z_c(\omega)$ and its low-order approximation

demonstrating the compensation technique of Robust TLNE model discussed in Section 2.4.6, $P_a(s)$ only gives 1st order, which results relatively higher deviations of phase angles in the high frequency range. The deviations due to reduced-order fitting is to be compensated by the deep region of TLNE. In addition, surface layer may also make up the deviations due to reduced-order deep region, especially in low frequency range such as DC and power frequency.

From individual lines which have the nodal equations (2.13a) and (2.13b), the admittance matrix of the original surface layer network is

$$\mathbf{Y}_{surface}(\omega) = \begin{bmatrix} \mathbf{Y}_{AA}(\omega) & \mathbf{Y}_{AB}(\omega) \\ \mathbf{Y}_{BA}(\omega) & \mathbf{Y}_{BB}(\omega) \end{bmatrix} \tag{2.41a}$$

and that of the reduced-order surface layer network

$$\tilde{\mathbf{Y}}_{surface}(\omega) = \begin{bmatrix} \tilde{\mathbf{Y}}_{AA}(\omega) & \tilde{\mathbf{Y}}_{AB}(\omega) \\ \tilde{\mathbf{Y}}_{BA}(\omega) & \tilde{\mathbf{Y}}_{BB}(\omega) \end{bmatrix} \tag{2.41b}$$

where subscript $_A$ stands for the ports connected to study zone, subscript $_B$ stands for those connected to deep region, and $\tilde{}$ designates simplification or approximation.

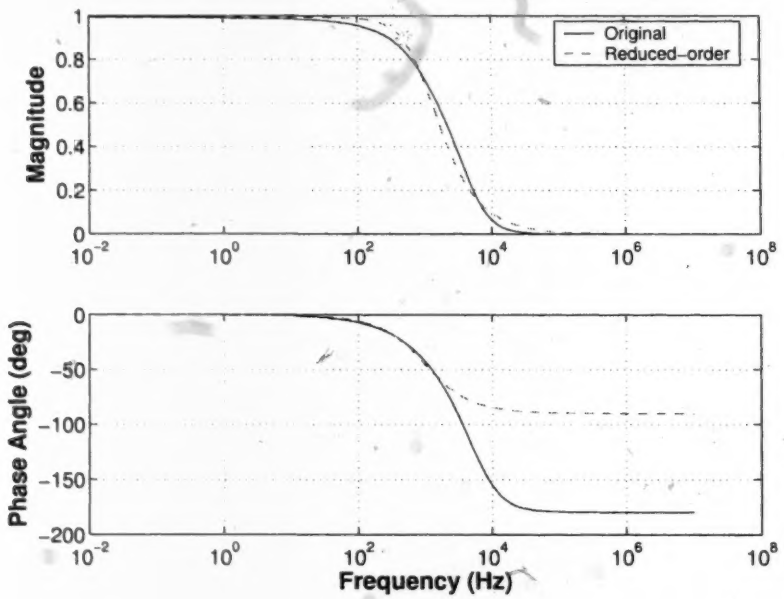


Figure 2.10: Frequency response of back-winded weighting function $P(\omega)$ and its low-order approximation

Since the external system is LTI, $\mathbf{Y}_{AB}(\omega) = [\mathbf{Y}_{BA}(\omega)]^T$ and $\tilde{\mathbf{Y}}_{AB}(\omega) = [\tilde{\mathbf{Y}}_{BA}(\omega)]^T$ where superscript ^{T} denotes transpose.

2.3 Deep Region

The fitting of external system by VF is stressed on relatively lower frequency range since high frequency transients do not travel very far in the external system. In TLNE, the deep region is further "insulated" from the study zone by the surface layer. Thus, the order of deep region may be significantly reduced. This gives the idea that the deep region can be represented by a low-order rational function (matrix) of FDNE model in (2.6), which not only represents the transient behavior of the original external system excluding the transmission network in surface layer, but also compensates for the differences due to the order reduction of transmission lines in surface layer. The deep region can be obtained by low-order VF approximation with emphasis on lower frequency range and Sequential Quadratic Programming (SQP) [21,22], or by the proposed robust approach which utilizes GAs and CLLSQ with optimal order selection introduced in

the subsequent sections.

2.4 Finding the First Approximations with GAs

The first approximation of external system input admittance [21,22], is the initial mathematical combination of admittance matrix $\tilde{\mathbf{Y}}_{surface}(\omega)$ of the surface layer constituting reduced-order lines and $\tilde{\mathbf{Y}}_{deep}(\omega)$ of deep region comprising low-order FDNE generated by VF. It is obtained from [21,22]

$$\tilde{\mathbf{Y}}^0_{input}(\omega) = \tilde{\mathbf{Y}}^0_{AA}(\omega) - \tilde{\mathbf{Y}}^0_{AB}(\omega)(\tilde{\mathbf{Y}}^0_{BB}(\omega) + \tilde{\mathbf{Y}}^0_{deep}(\omega))^{-1}\tilde{\mathbf{Y}}^0_{BA}(\omega) \tag{2.42}$$

where the superscript ^{0} denotes "first" since the subsequent optimizations (*e.g.*, SQP) are to be carried out; $\tilde{\mathbf{Y}}^0_{AA}(\omega)$, $\tilde{\mathbf{Y}}^0_{AB}(\omega)$, $\tilde{\mathbf{Y}}^0_{BA}(\omega)$ and $\tilde{\mathbf{Y}}^0_{BB}(\omega)$ corresponds to the blocks of the first approximation of surface layer admittance $\tilde{\mathbf{Y}}^0_{surface}(\omega)$ in (2.41b); the first approximation of deep region admittance $\tilde{\mathbf{Y}}^0_{deep}(\omega)$ has the form of (2.6), and $\tilde{\mathbf{Y}}^0_{input}(\omega)$ is the first approximation of external system input admittance. The ultimate goal of building TLNE is to match $\tilde{\mathbf{Y}}_{input}(\omega)$ with the original external system admittance $\mathbf{Y}_{input}(\omega)$ as close as possible while ensuring stability and passivity of the model and frequency response at DC and power frequency.

2.4.1 Problems in Finding the First Approximations

In frequency domain, compared to the original $\mathbf{Y}_{input}(\omega)$, the $\tilde{\mathbf{Y}}^0_{input}(\omega)$ will, to some extent, exhibit deviations. Reducing those deviations necessitates the further optimizations to be carried out. Due to the nonlinear nature of the parameters in surface layer and deep region with respect to $\mathbf{Y}_{input}(\omega)$, (2.42) must first be linearized with respect to the parameters to be optimized, and Nonlinear Programming (NLP) such as SQP is implemented since passivity criteria are also nonlinear constraints [21,22]. However, NLP method does not guarantee convergence and is computational expensive. Moreover, the first approximation of the deep region obtained by low-order VF with emphasis on lower frequency range is not able to ensure best choice of the deep region. This is because that both the selection of weighting function and that of order in VF are very arbitrary. In Example 1 in Section 2.6, VF produce pretty good results. However, VF fails to produce good first approximation for deep region in Example 2 in Section 3.3,

whose external system has very complicated frequency response. Thus, using VF alone can not always ensure a relatively good first approximation. It is necessary to find out alternative methods that are able to generate better first approximations for TLNE, if VF is unsuccessful to do so. A close-to-original first approximation also leads to only requiring a small fine-tuning in the subsequent optimizations, which can probably be fulfilled by least-square optimization instead of NLP. The least-square optimization is faster and more importantly, has guaranteed convergence.

2.4.2 Application of GAs

The idea for finding reduce-order deep region is based on the fact that in frequency domain, each resonant peak of original deep region admittance $\mathbf{Y}_{deep}(\omega)$ are likely to produce a resonance peak in the input admittance of external system $\mathbf{Y}_{input}(\omega)$. However, due to the insulation of surface layer, some peaks in $\mathbf{Y}_{deep}(\omega)$ are insensitive to $\mathbf{Y}_{input}(\omega)$. This gives the idea that removing some resonant peaks in deep region will impose little effect on $\mathbf{Y}_{input}(\omega)$. Thus, with the utilization of GAs, a better first approximation can be found by globally selecting the resonant peaks of the original deep region that have more significant effect on $\mathbf{Y}_{input}(\omega)$ in the designated low orders of deep region FDNE model.

This is a multi-variable optimization with nonlinear passivity constraint in single-port external systems and a multi-variable multi-objective optimization with the constraints in multi-port external systems. The GAs introduced in Section 2.1.6 are well-suited for solving the problem. Especially in case of multi-port external systems, the merits of GAs in multi-objective optimizations are fully utilized. Other methods such as Simulated Annealing (SA) are also able to solve this problem. However, in this thesis, GAs are applied. In fact, one of the most significant improvements in the robust approach compared to the existing one [21, 22] is the application of GAs in obtaining better first approximations. In order to apply GAs in global searching of best deep regions, data pre-processing steps must be carried out.

2.4.3 Data Preparations For GAs

The sum-of-partial-fraction form in (2.6) is desirable for GAs' global selection. First, the frequency response of the original deep region $\mathbf{Y}_{deep}(\omega)$ is obtained by frequency

scan in ATP [44], which is explained in Section 2.1.2. In order to stress on lower frequency range, during frequency scan, frequency points are logarithmically distributed from low frequency, *e.g.*, 10Hz, to a high frequency limit f_{max}. In common practice, the upper frequency limit f_{max} is where frequency response becomes virtually flat and constant, *e.g.*, 1MHz. It is however demonstrated later in Example 1 (Section 2.6) and Example 2 (Section 3.3) that due to the insulation of surface layer, resonant peaks of obtained deep region are mainly distributed in lower frequency range. Therefore, the upper frequency limit in frequency scan can be reduced to relatively lower frequencies, *e.g.*, 10kHz, which is demonstrated in the modeling of AIES in Section 4.4. The main advantage of lowering upper frequency limit is the lower order rational function (matrix) generated by full-order VF, which in turn reduces computational time of GAs and subsequent optimizations since unnecessary high-frequency resonant peaks in deep region are eliminated. With this method, the constant term(s) in (2.6) of the fitted rational function (matrix) may not be very accurate since the frequency response is not constant in the higher frequency range. Nonetheless, this can be compensated by surface layer during later optimizations.

In addition to the above frequency points in frequency scan, one extra frequency point at DC is considered, since DC offset is important in time-domain simulation as well. In ATP, frequency scan can not be done at DC, so a frequency point close to DC, *e.g.*, 10^{-10}Hz is supplied.

In the next step, the sum-of-partial-fraction form is obtained by applying VF to deep region admittance (matrix) $\mathbf{Y}_{deep}(\omega)$ in frequency domain with low RMS-error% to produce a high-order proper rational function (matrix) $\mathbf{Y}^{VF}_{deep}(s)$ in s-domain which is, in MIMO case, a matrix that shares a common set of poles. In this step, frequency point at DC is not included, since it is to be matched in the later optimizations. Moreover, the inclusion of this point will very likely generate rank deficiency warnings during using VF in MATLAB, which result in an undesirable $\mathbf{Y}^{VF}_{deep}(s)$. Then the $\mathbf{Y}^{VF}_{deep}(s)$ is to be fed into GAs.

2.4.4 GA Parameters and Flowchart

It is noticed that $\mathbf{Y}^{VF}_{deep}(s)$ commonly has both real poles and complex poles. In our experience, conforming the logarithmical distribution of frequency points, after adequate

iterations, VF only generates less than 6 real poles which belong to lower frequency range, *e.g.*, below 200Hz and all other complex poles which are commonly not in low frequency range. Therefore, the partial fractions of complex conjugate pairs are considered to be processed by GAs, whereas the ones of real poles are all chosen to be the partial fractions for deep region. Keeping the partial fractions of real poles is also very helpful in ensuring response at DC and power frequency with a little price of possibly increasing the deep region order by 2 to 4. Thus, each complex conjugate pair of partial fractions is treated as one entity and all pairs are indexed. Considering the searching of best-suited complex conjugate partial fractions, decision variables in GAs are designated to the integer indices of those partial fractions, by which chromosomes are encoded. To facilitate evolution process in GAs, real-valued representation is used, since it is a more efficient encoding in GAs [47]. Then the integers can be obtained by rounding the real-valued chromosomes. However, after rounding, some of the integers may appear as duplicate ones. To solve this problem, population initialization and mutation routines are extended for corrections to make sure all integers are not same with each other in every chromosome.

In this particular problem, GA parameters have to be properly set up for better efficiency. According to Section 2.1.6, the flowchart in Fig. 2.8 and the problem characteristics, the following guidelines are provided

- Population size is in accordance with the order of $\mathbf{Y}_{deep}^{VF}(s)$. In Example 1 in Section 2.6, the population size is 500 corresponding to $\mathbf{Y}_{deep}^{VF}(s)$'s order of 60. In Example 2 in Section 3.3, since the order of $\mathbf{Y}_{deep}^{VF}(s)$ is 220, the population size is at least 2000. The larger the population size generated, the faster the GAs will converge.

- Number of generations is not essential since other criteria dominantly control the termination of GAs. However, a too small number may lead to unacceptable results. In this problem, the number of generations should be at least 50.

- Generation gap is recommended to be the default value 0.9.

- Roulette Wheel Selection (RWS) is used since it more obeys the probabilistic rules than SUS [48].

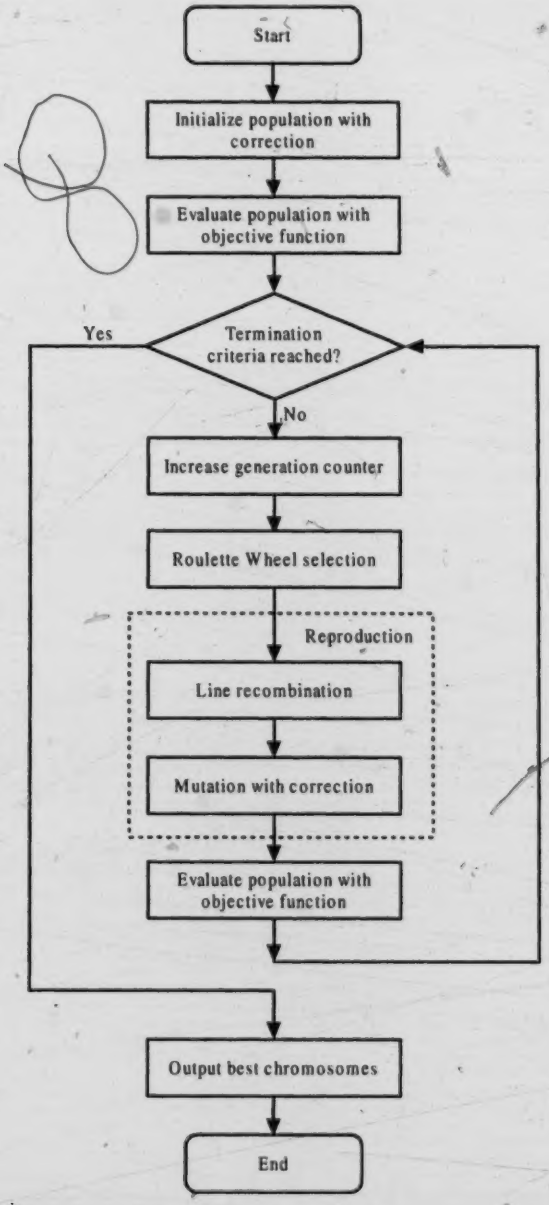


Figure 2.11: Flowchart of Genetic Algorithm for Robust TLNE

- Line recombination method with the probability of 0.8-0.95 is used. This is because the chromosomes are encoded by real numbers and the boundaries of selection scope must also be reached [48].

- Mutation with the probability of less than 0.7% is recommended.
- GA termination criteria are such emphasized that for a number of generations, the best value of objective function does not improve any more. A number between 30 to 50 is recommended.
- The best fitted individual generated by GAs does not guarantee the best deep region since further optimizations are to be carried out. Thus, the first N_{best} (*e.g.*, 5) number of best individuals are selected for future optimizations. Especially in multi-port system case, this strategy is more useful in finding better final results, which is examined in Example 2 in Section 3.3.

A detailed flowchart specific to this problem is shown in Fig. 2.11.

2.4.5 GA Objective Function

From above explanations on GAs' configurations, the admittance (SISO) or the elements of admittance matrix (MIMO) of the deep region $\tilde{\mathbf{Y}}^0_{deep}(\omega)$ generated by GAs have the following form

$$Y(s) = d + \sum_{i=1}^{n_r} \frac{c_i}{s - a_i} + \sum_{k=1}^{n_c/2} \left(\frac{c_k}{s - a_k} + \frac{c_k^*}{s - a_k^*} \right) \tag{2.43}$$

where for complex conjugate pairs,

$$a_k = a_k' + ja_k'',\ a_k^* = a_k' - ja_k'',\ c_k = c_k' + jc_k'',\ c_k^* = c_k' - jc_k'' \tag{2.44}$$

n_r is the number of real poles, n_c is that of complex poles, and the order of deep region $n_{deep} = n_r + n_c$. It is also noticed that $\tilde{\mathbf{Y}}^0_{deep}(\omega)$ still shares a common set of poles.

Calculation of $\tilde{\mathbf{Y}}^0_{input}(\omega)$ is based on the following equation

$$\tilde{\mathbf{Y}}^0_{input}(\omega) = \mathbf{Y}_{AA}(\omega) - \mathbf{Y}_{AB}(\omega)(\mathbf{Y}_{BB}(\omega) + \tilde{\mathbf{Y}}^0_{deep}(\omega))^{-1}\mathbf{Y}_{BA}(\omega) \tag{2.45}$$

Frequency points do not include DC, which will be optimized later. In (2.45), the surface layer is represented by the original admittance network $\mathbf{Y}_{surface}(\omega)$. This is because that in GAs, the search of the most significant resonant peaks is the goal, rather than compensate the simplified surface layer at same time. Both surface layer and deep region undergo further optimization later.

Since GAs try to find out the best low-order deep region $\tilde{\mathbf{Y}}^0_{deep}(\omega)$ that minimizes the difference between $\tilde{\mathbf{Y}}^0_{input}(\omega)$ and $\mathbf{Y}_{input}(\omega)$, the RMS-error or Frobenius Norm $\left\|\mathbf{Y}_{input}(\omega) - \tilde{\mathbf{Y}}^0_{input}(\omega)\right\|_F^2$ defines one part of the objective function. In addition, GAs generated $\tilde{\mathbf{Y}}^0_{deep}(\omega)$ must be positive-real. Thus the objective function is defined as [23]

$$f_{obj} = \left\|\mathbf{Y}_{input}(\omega) - \tilde{\mathbf{Y}}^0_{input}(\omega)\right\|_F^2 + \mu = \sum_{i,j=1}^{m} \left|\mathbf{Y}_{input,ij}(\omega) - \tilde{\mathbf{Y}}^0_{input,ij}(\omega)\right|^2 + \mu \quad (2.46)$$

where $\mathbf{Y}_{input,ij}(\omega)$ is the ij-th elements of $\mathbf{Y}_{input}(\omega)$; μ denotes a penalty term when passivity criterion violation occurs in deep region. If this criterion is violated, μ will a big enough positive number, otherwise $\mu = 0$. This ensures that the outputs from GAs are the best fitted deep regions, which are both stable and passivity. For passivity violation case, the value of μ is such a big value that individuals with the violation are very likely to be eliminated during selection step in later generations. In our experience, the value of 0.1 to 0.5 is used since the RMS-error% is not expected to be very high.

2.4.6 Compensation Technique for GAs

Removing partial fractions from $\tilde{\mathbf{Y}}^{VF}_{deep}(s)$ may result in the system frequency response especially in lower frequency range showing pronounced deviation from $\mathbf{Y}_{deep}(\omega)$. Assuming the following n_{cs} number of partial fractions $\sum_{k=1}^{n_{cs}/2}\left(\frac{c_k}{s-a_k} + \frac{c_k^*}{s-a_k^*}\right)$ are chosen by GAs, *i.e.*, the number of $n_c - n_{cs}$ partial fractions $\sum_{i=1}^{(n_c-n_{cs})/2}\left(\frac{c_i}{s-a_i} + \frac{c_i^*}{s-a_i^*}\right)$ are deselected for deep region, we have the elements of deep region admittance matrix as

$$\tilde{Y}(s) = d + \sum_{i=1}^{n_r}\frac{c_i}{s-a_i} + \sum_{k=1}^{n_{cs}/2}\left(\frac{c_k}{s-a_k} + \frac{c_k^*}{s-a_k^*}\right) \quad (2.47)$$

where a_k and c_k have the same meanings as in (2.44). Then the deviations with respect to full-order $Y(\omega)$ in frequency response are most significant in the lower frequency range and have the DC value of

$$Y_{dc} = -\sum_{k=1}^{(n_c-n_{cs})/2}\left(\frac{2(a'_k c'_k + a''_k c''_k)}{(a'_k)^2 + (a''_k)^2}\right) \quad (2.48)$$

which will deteriorate the corresponding $\tilde{\mathbf{Y}}^0_{input}(\omega)$. This can be compensated by adding one extra partial fraction of a real pole and a real residue $\frac{c_{kc}}{s-a_{kc}}$ where a_{kc} and c_{kc}

are [23]

$$a_{kc} = \max(a_k''), \quad 1 \leq k \leq n_{cs} \tag{2.49}$$

$$c_{kc} = -a_{kc} Y_{dc} \tag{2.50}$$

By adding one more order to the deep region, deviations resulted from removing partial fractions are minimized. Moreover, this partial fraction also increases the flexibilities of later CLLSQ optimizations. It is observed that the partial fraction may not be positive-real. However, this is not considered as an issue since the generated $\tilde{\mathbf{Y}}^0_{deep}(\omega)$ is positive-real.

2.4.7 Building the First Approximation of Input Admittance

After $\tilde{\mathbf{Y}}^0_{surface}(\omega)$ and $\tilde{\mathbf{Y}}^0_{deep}(\omega)$ are obtained, the first approximation of external system input admittance is generated through (2.42). Notice that in building the first approximation, reduced-order transmission line models in surface layer $\tilde{\mathbf{Y}}^0_{surface}(\omega)$ are used. Since deep regions generated by GAs are multiple, a series of first approximations of input admittance are built.

2.5 Constrained Linearized Least-Square Optimization

Theoretically, the generated input admittance $\tilde{\mathbf{Y}}^0_{input}(\omega)$ from Section 2.4 is very close to the original $\mathbf{Y}_{input}(\omega)$. Nonetheless, reduced-order surface layer $\tilde{\mathbf{Y}}^0_{surface}(\omega)$ and GAs generated deep region $\tilde{\mathbf{Y}}^0_{deep}(\omega)$ are subject to further fine-tunings or optimizations to minimize the deviations between $\tilde{\mathbf{Y}}_{input}(\omega)$ and $\mathbf{Y}_{input}(\omega)$. Frequency points include DC since frequency response at DC is to be matched during optimization.

2.5.1 Parameters Subject to Optimization

Parameters eligible for optimization include the following:

- In surface layer, k_0 terms, all poles p_i and residues r_i of $Z_{eq}(s)$ in (2.22); all poles p_i and residues r_i of $P_a(s)$ in (2.32); propagation constants τ.
- In deep region, all constant terms d_{ik}, poles a_i and residues c_{ik} in (2.6), the elements of which have the form of (2.43).

However, the propagation constants τ represents the length of the line and not recommended for changing, so all τ in surface layer are not chosen for optimization. Our experience shows that modifying the poles shared by all elements of deep region admittance matrix does not help to improve $\tilde{\mathbf{Y}}_{input}(\omega)$ with respect to $\mathbf{Y}_{input}(\omega)$, rather, sometimes optimizing those poles may cause larger RMS-error% with respect to the cases without optimization of those poles. Constructing surface layer admittance matrix from Marti's line model parameters also shows that the partial derivatives with respect to parameters of $P_a(s)$ are very complicated, which is more likely to cause convergence problems and result in larger RMS-error% in optimization than other parameters. So the parameters of $Z_{eq}(s)$ are more suitable for optimization. Therefore, parameters considered for further optimization are

- In surface layer, k_0 terms, all poles p_i and residues r_i of $Z_{eq}(s)$ in (2.22).
- In deep region, all constant terms d_{ik} and residues c_{ik} in (2.6), the elements of which have the form of (2.43).

It has been pointed out in [21,22] that optimization of surface layer parameters are considered only a last resort for accuracy. However, in our experience, since it particularly improves the frequency response at DC and power frequency, optimizing surface layer parameters is chosen by default in the Robust TLNE.

2.5.2 Linearization

Equation (2.42) expressed in s-domain as

$$\tilde{\mathbf{Y}}_{input}(s) = \tilde{\mathbf{Y}}_{AA}(s) - \tilde{\mathbf{Y}}_{AB}(s)(\tilde{\mathbf{Y}}_{BB}(s) + \tilde{\mathbf{Y}}_{deep}(s))^{-1}\tilde{\mathbf{Y}}_{BA}(s) \tag{2.51}$$

is linearized in frequency domain with respect to parameters of surface layer and deep region at the point of $\tilde{\mathbf{Y}}^0_{input}(\omega)$ to produce [21,22]

$$\mathbf{Y}_{input}(\omega) = \tilde{\mathbf{Y}}^0_{input}(\omega) + \mathbf{J}(\omega)\Delta\mathbf{x} \tag{2.52}$$

i.e.,

$$\Delta\mathbf{Y}_{input}(\omega) = \mathbf{Y}_{input}(\omega) - \tilde{\mathbf{Y}}^0_{input}(\omega) = \mathbf{J}(\omega)\Delta\mathbf{x} \tag{2.53}$$

with

$$\mathbf{x} = \mathbf{x}_0 + \Delta\mathbf{x} \tag{2.54}$$

where $\tilde{\mathbf{Y}}^0_{input}(\omega)$ stands for the first approximation of input admittance matrix discussed in last section, $\mathbf{J}(\omega)$ is the Jacobian matrix, $\mathbf{x}$ is the model parameter column vector considered for optimization and $\Delta\mathbf{x}$ is the model parameter change column vector. The overdetermined linear equation (2.53) shows a least-square sense, which implies that further improvements of $\tilde{\mathbf{Y}}^0_{input}(\omega)$ with respect to $\mathbf{Y}_{input}(\omega)$ can be obtained through recursive evaluation of (2.53) with (2.54).

2.5.3 The Jacobian Matrix

To evaluate the Jacobian Matrix, partial derivatives with respect to parameters to be optimized are obtained first. From (2.51), in surface layer, the partial derivatives with respect to a real parameter ρ are given as [21, 22]

$$\begin{aligned}\frac{\partial\tilde{\mathbf{Y}}_{input}(s)}{\partial\rho} = {} & \frac{\partial\tilde{\mathbf{Y}}_{AA}(s)}{\partial\rho} - \frac{\partial\tilde{\mathbf{Y}}_{AB}(s)}{\partial\rho}[\tilde{\mathbf{Y}}_{BB}(s)+\tilde{\mathbf{Y}}_{deep}(s)]^{-1}\tilde{\mathbf{Y}}_{BA}(s) \\ & - \tilde{\mathbf{Y}}_{AB}(s)[\tilde{\mathbf{Y}}_{BB}(s)+\tilde{\mathbf{Y}}_{deep}(s)]^{-1}\frac{\partial\tilde{\mathbf{Y}}_{BA}(s)}{\partial\rho} \\ & + \tilde{\mathbf{Y}}_{AB}(s)[\tilde{\mathbf{Y}}_{BB}(s)+\tilde{\mathbf{Y}}_{deep}(s)]^{-1}\frac{\partial\tilde{\mathbf{Y}}_{BB}(s)}{\partial\rho}[\tilde{\mathbf{Y}}_{BB}(s)+\tilde{\mathbf{Y}}_{deep}(s)]^{-1}\tilde{\mathbf{Y}}_{BA}(s)\end{aligned} \tag{2.55}$$

Consider a transmission line connecting between Bus N_1 and Bus N_2 in surface layer which has parameters of τ, $Z_{eq}(s)$ and $P_a(s)$ in s-domain, the elements of its admittance matrix $\mathbf{Y}_{TL}(s)$ in surface layer are given as

$$(\mathbf{Y}_{TL}(s))_{ij} = \begin{cases} -\dfrac{e^{-\tau s}}{Z_{eq}(s)}\left(\dfrac{P_a(s)+P_a^{-1}(s)}{P_a(s)-P_a^{-1}(s)}\right) & \text{when } i=j=N_1 \text{ or } i=j=N_2 \\ \dfrac{e^{-\tau s}}{Z_{eq}(s)}\left(\dfrac{2}{P_a(s)-P_a^{-1}(s)}\right) & \text{when } i=N_1, j=N_2 \text{ or } i=N_2, j=N_1 \\ 0 & \text{otherwise}\end{cases} \tag{2.56}$$

Thus, partial derivatives of $\tilde{\mathbf{Y}}_{surface}(s)$ with respect to the parameters of a specific line $Z_{eq}(s)$ in (2.22) are obtained, which is shown in Table 2.1. $\partial\tilde{\mathbf{Y}}_{AA}(s)/\partial\rho$, $\partial\tilde{\mathbf{Y}}_{AB}(s)/\partial\rho$, $\partial\tilde{\mathbf{Y}}_{BA}(s)/\partial\rho$ and $\partial\tilde{\mathbf{Y}}_{BB}(s)/\partial\rho$ in (2.55) are given by partitioning $\partial\tilde{\mathbf{Y}}_{surface}(s)/\partial\rho$ according to the buses connecting to study zone (subscript $_A$) and those connecting to deep region (subscript $_B$).

Parameter ρ in $Z_{eq}(s)$	Partial derivative $\partial\tilde{\mathbf{Y}}_{surface}(s)/\partial\rho$
k_0	$-\frac{1}{Z_{eq}(s)}\mathbf{Y}_{TL}(s)$
r_i	$-\frac{1}{(s+p_i)Z_{eq}(s)}\mathbf{Y}_{TL}(s)$
p_i	$\frac{r_i}{(s+p_i)^2 Z_{eq}(s)}\mathbf{Y}_{TL}(s)$

Table 2.1: Partial derivatives with respect to parameters of $Z_{eq}(s)$ in surface layer

Partial derivatives with respect to deep region parameters are obtained by [21,22]

$$\frac{\partial\tilde{\mathbf{Y}}_{input}(s)}{\partial\rho} = \tilde{\mathbf{Y}}_{AB}(s)[\tilde{\mathbf{Y}}_{BB}(s)+\tilde{\mathbf{Y}}_{deep}(s)]^{-1}\frac{\partial\tilde{\mathbf{Y}}_{deep}(s)}{\partial\rho}[\tilde{\mathbf{Y}}_{BB}(s)+\tilde{\mathbf{Y}}_{deep}(s)]^{-1}\tilde{\mathbf{Y}}_{BA}(s) \tag{2.57}$$

Parameter ρ in the N-th diagonal elements of $\tilde{\mathbf{Y}}_{deep}(s)$	The ij-th matrix elements of partial derivative $\partial\tilde{\mathbf{Y}}_{deep}(s)/\partial\rho$
d	$\begin{cases} 1 & \text{when } i=j=N \\ 0 & \text{otherwise} \end{cases}$
c_k	$\begin{cases} \frac{1}{s-a_k} & \text{when real and } i=j=N \\ f_k(s) & \text{when complex and } i=j=N \\ 0 & \text{otherwise} \end{cases}$

Table 2.2: Partial derivatives with respect to parameters of diagonal elements in deep region admittance matrix

!h However, for deep region defined by (2.6), in order to find out $\partial\tilde{\mathbf{Y}}_{deep}(s)/\partial\rho$, diagonal and off-diagonal elements in (2.6) have to be considered separately. Partial derivatives are with respect to real parameters. Therefore, the complex conjugate pairs in (2.43) are considered as real and imaginary parts shown in (2.44). For the N-th diagonal element, partial derivatives of $\tilde{\mathbf{Y}}_{deep}(s)$ with respect to a real parameter ρ are shown

in Table 2.2, where $f_k(s)$ is defined as

$$f_k(s) = \begin{cases} \dfrac{1}{s-a_k} + \dfrac{1}{s-a_k^*} & \text{for real part } c' \\ j\left(\dfrac{1}{s-a_k} - \dfrac{1}{s-a_k^*}\right) & \text{for imaginary part } c'' \end{cases} \tag{2.58}$$

Partial derivatives of $\tilde{\mathbf{Y}}_{deep}(s)$ with respect to parameters of off-diagonal elements are very similar to diagonal ones except that two equal elements exist in the matrix due to the symmetry of $\tilde{\mathbf{Y}}_{deep}(s)$. For an off-diagonal element at N_1 row and N_2 column, partial derivatives $\partial\tilde{\mathbf{Y}}_{deep}(s)/\partial\rho$ are given in Table 2.3, where $f_k(s)$ is also defined by (2.58).

Parameter ρ in the N_1N_2-th off-diagonal elements of $\tilde{\mathbf{Y}}_{deep}(s)$	The ij-th matrix elements of partial derivative $\partial\tilde{\mathbf{Y}}_{deep}(s)/\partial\rho$
d	$\begin{cases} 1 & \text{when } i=N_1, j=N_2 \text{ or } i=N_2, j=N_1 \\ 0 & \text{otherwise} \end{cases}$
c_k	$\begin{cases} \dfrac{1}{s-a_k} & \text{when real and } i=N_1, j=N_2 \text{ or } i=N_2, j=N_1 \\ f_k(s) & \text{when complex and } i=N_1, j=N_2 \text{ or } i=N_2, j=N_1 \\ 0 & \text{otherwise} \end{cases}$

Table 2.3: Partial derivatives with respect to parameters of off-diagonal elements in deep region admittance matrix

With the partial derivatives formulated above, the Jacobian matrix in s-domain is formed corresponding to the parameters to be optimized as

$$\mathbf{J}_1(s) = \left[\frac{\partial\tilde{\mathbf{Y}}_{input}(s)}{\partial\rho_1} \quad \frac{\partial\tilde{\mathbf{Y}}_{input}(s)}{\partial\rho_2} \quad \cdots \frac{\partial\tilde{\mathbf{Y}}_{input}(s)}{\partial\rho_{N_p}} \right] \tag{2.59}$$

where N_p is the total number of parameters to be optimized. However, in multi-port external system case, each matrix $\partial\tilde{\mathbf{Y}}_{input}(s)/\partial\rho_k$ $(1 \le k \le N_p)$ must be expanded to obtain the form suitable for least-square sense. Because of the symmetrical nature of the external system admittance matrix, only the upper or lower triangular parts are expended, the purpose of which is to reduce computational burdens. Therefore, we

have

$$\mathbf{J}(s) = \begin{bmatrix} \left(\frac{\partial \tilde{\mathbf{Y}}_{input}(s)}{\partial \rho_1}\right)_{11} & \left(\frac{\partial \tilde{\mathbf{Y}}_{input}(s)}{\partial \rho_2}\right)_{11} & \cdots & \left(\frac{\partial \tilde{\mathbf{Y}}_{input}(s)}{\partial \rho_{N_p}}\right)_{11} \\ \left(\frac{\partial \tilde{\mathbf{Y}}_{input}(s)}{\partial \rho_1}\right)_{12} & \left(\frac{\partial \tilde{\mathbf{Y}}_{input}(s)}{\partial \rho_2}\right)_{12} & \cdots & \left(\frac{\partial \tilde{\mathbf{Y}}_{input}(s)}{\partial \rho_{N_p}}\right)_{12} \\ \vdots & \vdots & \ddots & \vdots \\ \left(\frac{\partial \tilde{\mathbf{Y}}_{input}(s)}{\partial \rho_1}\right)_{N_{es}N_{es}} & \left(\frac{\partial \tilde{\mathbf{Y}}_{input}(s)}{\partial \rho_2}\right)_{N_{es}N_{es}} & \cdots & \left(\frac{\partial \tilde{\mathbf{Y}}_{input}(s)}{\partial \rho_{N_p}}\right)_{N_{es}N_{es}} \end{bmatrix} \tag{2.60}$$

where N_{es} is the total number of elements in upper/lower triangular part of external system admittance matrix, which has the same dimension of its partial derivative matrices. Since frequency points are discrete, (2.60) is further evaluated in frequency domain

at discrete frequency points and stacked to form the final Jacobian matrix as

$$\mathbf{J}(\omega)' =$$

$$\left[\begin{array}{ccc}
\begin{pmatrix}\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_1)}{\partial\rho_1}\\ \frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_2)}{\partial\rho_1}\end{pmatrix}_{11} & \begin{pmatrix}\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_1)}{\partial\rho_2}\\ \frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_2)}{\partial\rho_2}\end{pmatrix}_{11} & \cdots \begin{pmatrix}\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_1)}{\partial\rho_{N_p}}\\ \frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_2)}{\partial\rho_{N_p}}\end{pmatrix}_{11} \\
\vdots & \vdots & \ddots \quad \vdots \\
\left(\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_{N_f})}{\partial\rho_1}\right)_{11} & \left(\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_{N_f})}{\partial\rho_2}\right)_{11} & \cdots \left(\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_{N_f})}{\partial\rho_{N_p}}\right)_{11} \\
\hline
\begin{pmatrix}\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_1)}{\partial\rho_1}\\ \frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_2)}{\partial\rho_1}\end{pmatrix}_{12} & \begin{pmatrix}\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_1)}{\partial\rho_2}\\ \frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_2)}{\partial\rho_2}\end{pmatrix}_{12} & \cdots \begin{pmatrix}\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_1)}{\partial\rho_{N_p}}\\ \frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_2)}{\partial\rho_{N_p}}\end{pmatrix}_{12} \\
\vdots & \vdots & \ddots \quad \vdots \\
\left(\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_{N_f})}{\partial\rho_1}\right)_{12} & \left(\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_{N_f})}{\partial\rho_2}\right)_{12} & \cdots \left(\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_{N_f})}{\partial\rho_{N_p}}\right)_{12} \\
\hline
\vdots & \vdots & \ddots \quad \vdots \\
\hline
\begin{pmatrix}\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_1)}{\partial\rho_1}\\ \frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_2)}{\partial\rho_1}\end{pmatrix}_{N_{es}N_{es}} & \begin{pmatrix}\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_1)}{\partial\rho_2}\\ \frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_2)}{\partial\rho_2}\end{pmatrix}_{N_{es}N_{es}} & \cdots \begin{pmatrix}\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_1)}{\partial\rho_{N_p}}\\ \frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_2)}{\partial\rho_{N_p}}\end{pmatrix}_{N_{es}N_{es}} \\
\vdots & \vdots & \ddots \quad \vdots \\
\left(\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_{N_f})}{\partial\rho_1}\right)_{N_{es}N_{es}} & \left(\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_{N_f})}{\partial\rho_2}\right)_{N_{es}N_{es}} & \cdots \left(\frac{\partial\tilde{\mathbf{Y}}_{input}(\omega_{N_f})}{\partial\rho_{N_p}}\right)_{N_{es}N_{es}}
\end{array}\right] \tag{2.61}$$

where N_f is the number of discrete frequency points.

2.5.4 Iterative Least-Square Optimization

To apply least-square optimization, in multi-port external system case, all elements of the original input admittance matrix $\mathbf{Y}_{input}(\omega)$ and the calculated $\tilde{\mathbf{Y}}_{input}(\omega)$ of each iteration in (2.53) are also evaluated at the same frequency points as the Jacobian matrix and stacked to form a column vector as

$$\begin{aligned}\Delta\mathbf{Y}_{input}(\omega) = \\ \big[(\Delta\mathbf{Y}_{input}(\omega_1))_{11} \quad (\Delta\mathbf{Y}_{input}(\omega_2))_{11} \quad \cdots \quad (\Delta\mathbf{Y}_{input}(\omega_{N_f}))_{11} \\ (\Delta\mathbf{Y}_{input}(\omega_1))_{12} \quad (\Delta\mathbf{Y}_{input}(\omega_2))_{12} \quad \cdots \quad (\Delta\mathbf{Y}_{input}(\omega_{N_f}))_{12} \\ (\Delta\mathbf{Y}_{input}(\omega_1))_{N_{es}N_{es}} \quad (\Delta\mathbf{Y}_{input}(\omega_2))_{N_{es}N_{es}} \quad \cdots \quad (\Delta\mathbf{Y}_{input}(\omega_{N_f}))_{N_{es}N_{es}}\big]^T\end{aligned} \tag{2.62}$$

Also due to the symmetry of $\Delta\mathbf{Y}_{input}(\omega)$, the same method as stacking the Jacobian matrix is applied. In addition, the sequence of stacking $\Delta\mathbf{Y}_{input}(\omega)$ must be in accordance with that of stacking the Jacobian matrix.

Thus, from (2.53), (2.61) and (2.62), an overdetermined linear equation $\Delta\mathbf{Y}_{input}(\omega) = \mathbf{J}(\omega)\Delta\mathbf{x}$ of finding $\Delta\mathbf{x}$ to minimize $\Delta\mathbf{Y}_{input}(\omega)$ in the least-square sense is constructed. However, the least-square sense does not apply to complex quantities. Nonetheless, in (2.53), $\Delta\mathbf{Y}_{input}(\omega)$ and $\mathbf{J}(\omega)$ are complex quantities. Therefore, real and imaginary parts are separated and stacked to produce

$$\begin{bmatrix} \mathrm{Re}(\Delta\mathbf{Y}_{input}(\omega)) \\ \mathrm{Im}(\Delta\mathbf{Y}_{input}(\omega)) \end{bmatrix} = \begin{bmatrix} \mathrm{Re}(\mathbf{J}(\omega)) \\ \mathrm{Im}(\mathbf{J}(\omega)) \end{bmatrix} \Delta\mathbf{x} \tag{2.63}$$

During the optimization, both positive-real criterion and algorithm convergence must be guaranteed, which leads to the concept of constrained optimizations. It has been discussed in [21, 22] that the NLP methods such as SQP are required to accomplish this task. Consequently, (2.63) has to be converted to the more complex quadratic form. However, the NLP methods are prone to divergence and are computationally expensive. In the Robust TLNE, since GAs are able to obtain better first approximations that are both stable and passive, *i.e.*, the first approximations are already in the feasible region in optimization sense and close to original, only minor fine-tunings are required. Therefore, instead of NLP, Constrained Linearized Least-Square (CLLSQ) optimization is employed.

In each iteration of evaluating $\Delta\mathbf{x}$, both stability and passivity conditions for surface layer and deep region are checked after adding parameter changes. Checking

positive-real criterion for $\tilde{\mathbf{Y}}_{input}(\omega)$ is not necessary since the surface layer and deep region belongs to different models. As far as both surface layer and deep region are stable and positive-real, stable time-domain simulations are guaranteed. During the condition check, each transmission line in surface layer and the whole deep region are treated as separate entities. The entities with either condition violation result a positive number δ ($0 < \delta < 1$) is multiplied with the entities' parameters change (a part of $\Delta\mathbf{x}$), since they have changed too much. Another essential condition in the optimization is the decrease of RMS-error% in the model input admittance during subsequent iterations. If RMS-error% does not decrease compared to the last iteration, the whole parameter change vector $\Delta\mathbf{x}$ is also multiplied by δ. The optimal point or termination criterion is that for a consecutive number N_{op} times of multiplication of δ with $\Delta\mathbf{x}$, the RMS-error% of input admittance does not decrease. Therefore, (2.54) is re-written as

$$\mathbf{x} = \begin{cases} \mathbf{x_0} + (\delta)^q \Delta\mathbf{x_1} + \Delta\mathbf{x_2} & \text{when PR or stability violations,} \\ \mathbf{x_0} + (\delta)^q \Delta\mathbf{x} & \text{when larger RMS-error,} \\ \mathbf{x_0} + \Delta\mathbf{x} & \text{otherwise (no violations).} \end{cases} \tag{2.64}$$

where $0 < q < N_{op}$, $\Delta\mathbf{x_1}$ are parameter changes of criterion violation and $\Delta\mathbf{x_2}$ are the parameters not violating criteria. δ is recommended for values from 0.1 to 0.5. The value of 5 to 10 for N_{op} is recommended.

2.5.5 Optimal Deep Region Order Determination

So far, the discussion in the Robust TLNE is only limited to a specific deep region order. However, to obtain the best suitable order of deep region, a number of deep region orders are applied to this problem. The intuitive idea is to construct a loop from n_{low} (n_{low} is greater than the number of real partial fractions) to n_{high} (n_{high} is less than the order of $\tilde{\mathbf{Y}}^{VF}_{deep}$) for deep region orders and find the deep region with lowest RMS-error% in each iteration of loops. Since GAs only choose complex conjugate pairs of deep region, in the loop, the order increases by two. Thus we obtain the order of deep region *v.s.* RMS-error% from n_{low} to n_{high}. The acceptable RMS-error% range is below 10% in our practice. Within this range, the optimal order is the one where in the orders lower than it, the RMS-error% increases dramatically, whereas in the orders higher than it, the RMS-error% does not decrease significantly. The later examples and case study illustrate this idea. In Example 1, Fig. 2.13 shows that the order of 13 for deep region are

the optimal order. In Example 2, 21 is the best suitable order for deep region in TLNE model.

2.6 Example 1

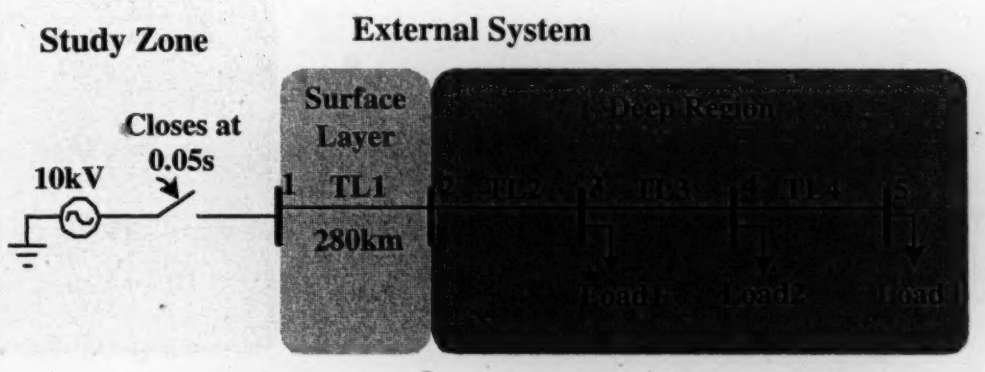


Figure 2.12: Example 1 system diagram and its partitioning

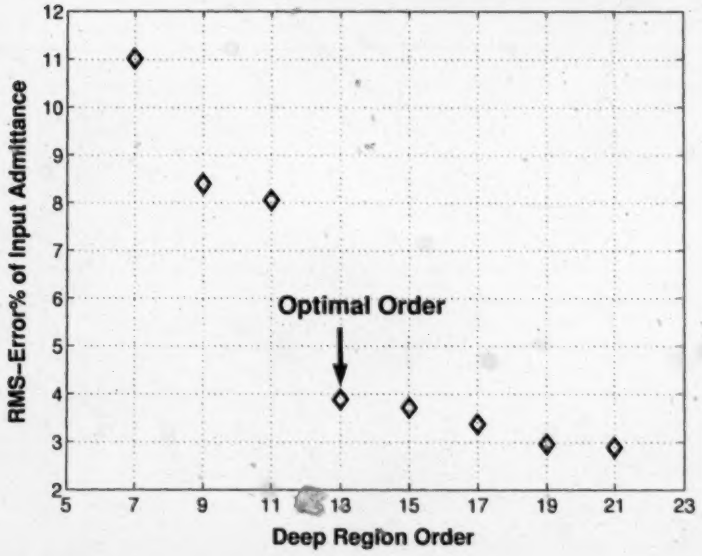


Figure 2.13: Example 1 RMS-error% of input admittance versus deep region order

Example 1, whose external system is a single-phase single-port passive network, shown in Fig. 2.12, is a transmission line energization case. Detailed parameters for the

system are listed in Appendix A.1. The same Drake conductor and tower configuration are assumed in all transmission lines. A 60Hz 10kV ideal voltage source is switched to the passive part after 0.05s. Switching current is measured at Bus 1. Fig. 2.12 also shows partitioning the system into a study zone and an external system.

2.6.1 Generation of the Robust TLNE

Also shown in Fig. 2.12, the external system is further partitioned into a surface layer and a deep region. The surface layer consists of only one 280km transmission line TL1, since the deep region is less complex. The original deep region admittance (shown in Fig. 2.15) is fitted by a 40th-order rational function in s-domain with 0.346% RMS-error, which has 4 real poles all below 200Hz and 18 complex conjugate pole pairs. Then, the 18 pairs of complex conjugates are processed by GAs. Fig. 2.13 shows RMS-error% *v.s.* deep region order ranging from 7 to 21. It has been clearly identified that 13 is optimal order for deep region, which has the RMS-error of 3.893% in input admittance after CLLSQ optimization. At DC, the input admittance only has 0.0012% deviation. With 13th order for deep region, Fig. 2.15 shows the deep region frequency response generated by GAs and after CLLSQ optimization. Some resonant peaks in deep region are not selected due to their insensitivity to input admittance. On the other hand, the low-order deep region catches some major peaks of the original frequency response. Moreover, CLLSQ also compensates the deviations due to reduced-order surface layer models. Fig. 2.14 shows TL1 characteristic impedance in surface layer of reduced order and after CLLSQ optimization. It is observed that the change of surface layer parameters during CLLSQ helps to increase the accuracy of Robust TLNE. The resultant input admittance of Robust TLNE model, shown in Fig. 2.16, is very accurate. It is also noticed that by applying GAs, the first approximation of input admittance is already close to original. This verifies the accuracy of GAs in obtaining optimal deep region. Appendix A.3 and A.4 lists ATP data file of full system and Robust TLNE model, respectively.

2.6.2 Transient Simulations

Shown in Fig. 2.17, transient simulation of Bus 1 branch current at 10μs step size further verifies the indistinguishability between full model and Robust TLNE model. The total

Transient event	Simulation time	Full model	Robust TLNE	FDNE(18th)
Line energization	0.2s	0.173s	0.061s	0.092s

Table 2.4: Example 1 computational time comparison at time-step size 10μs

simulation time T_{max} is 0.2s in this case. In order to emphasize the transient, only simulation time from 0.03s to 0.15s is displayed. In a Pentium IV 1.6GHz computer, the simulation of full model in ATP requires 0.173s, whereas the simulation of Robust TLNE model needs 0.061s to accomplish, which is about almost three times of save on computational time. For a similar RMS-error% FDNE model for the external system, VF generates an 18th-order rational function of 3.774% RMS-error, which requires 0.092s for the whole simulation. This is slightly higher than the one of Robust TLNE model. Table 2.4 shows simulation time comparison for different external system models. In a small system with less complex frequency response, the Robust TLNE model does not demonstrate significant computational save with respect to FDNE model. Later in Example 2, a much bigger computational save with high accuracy is observed. With existing TLNE approach, at 13th order for deep region, the model achieves 4.629% RMS-error on input admittance with respect to the original, which is slightly higher that the proposed approach. However, in Example 2, the RMS-error% of existing TLNE model is much higher than that of Robust TLNE model.

2.7 Summary

In this chapter, with discussions on some basic models, concepts and methods, the building of a Robust TLNE model for passive networks is explained in detail, . The external system is divided into a surface layer of reduced-order frequency-dependent transmission lines and a deep region of low-order FDNE model. GAs are applied to find best suitable deep region and CLLSQ optimization is used to fine-tune model parameters and further improve accuracy including DC frequency. One single-phase single-port example with a passive external system is presented. Comparisons between full model, Robust TLNE and FDNE are made in terms of computational time and accuracy. Transient simulation result verifies the accuracy of the Robust TLNE model and simu-

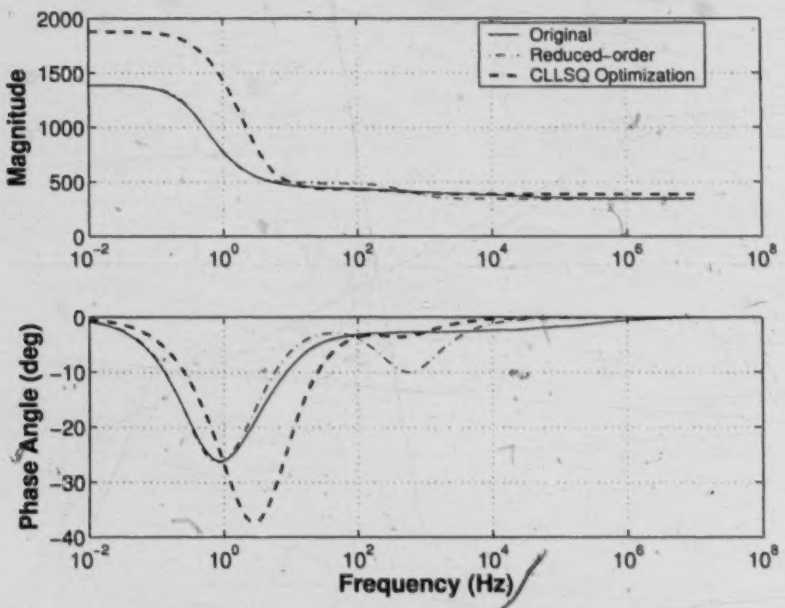


Figure 2.14: Example 1 TL1 characteristic impedance

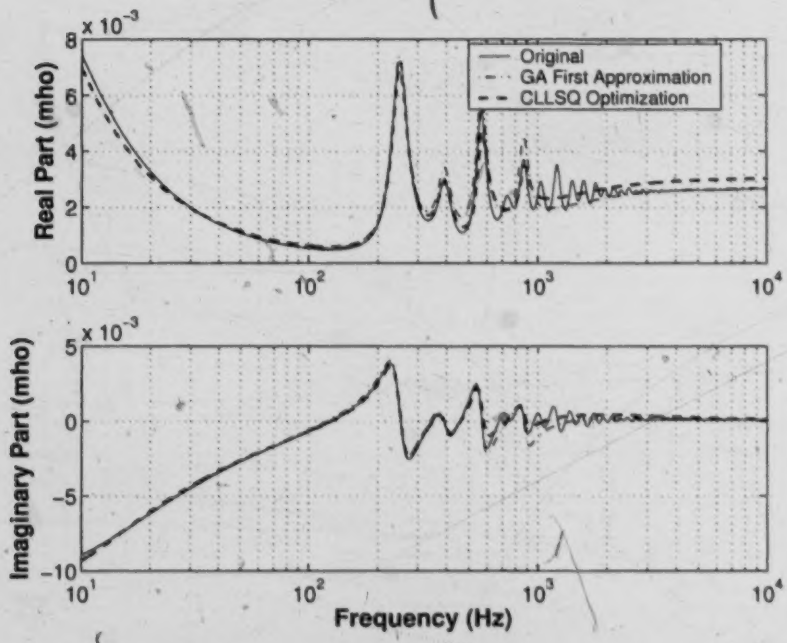


Figure 2.15: Example 1 deep region admittance

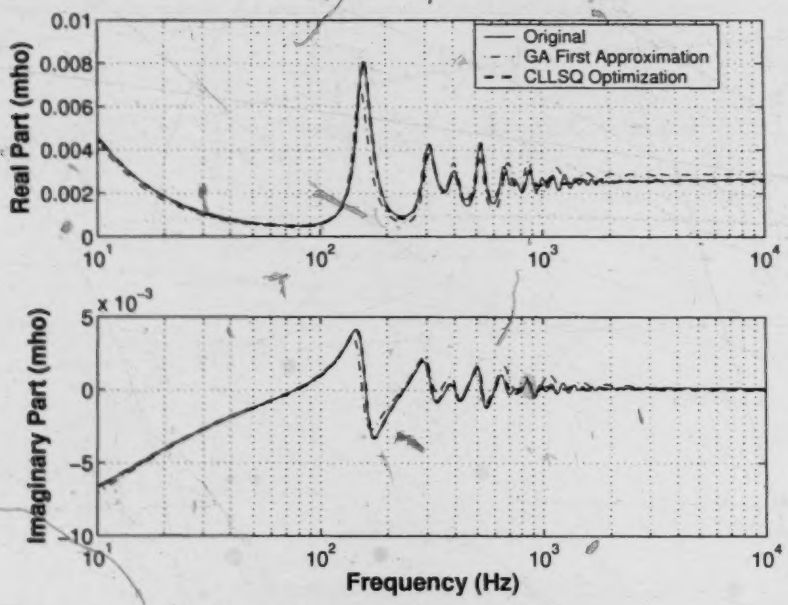


Figure 2.16: Example 1 input admittance

lation time analysis demonstrates computational burden reduction. In the next chapter, the Robust TLNE model is extended to active and multi-phase networks.

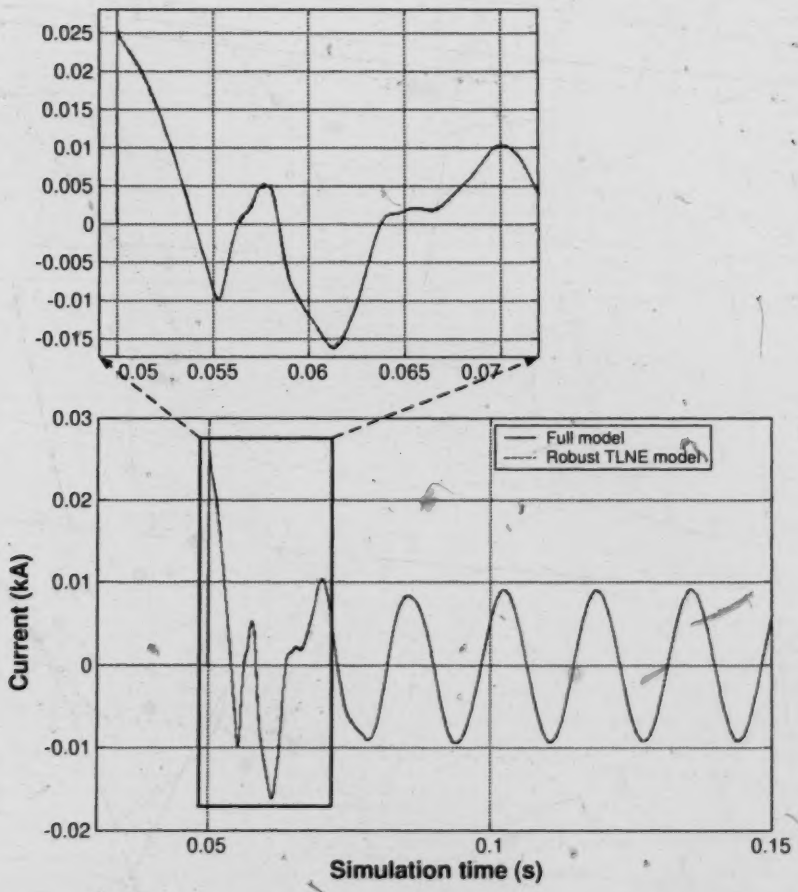


Figure 2.17: Example 1 transient simulation and comparison

3

Robust Two-Layer Network Equivalent for Active Networks

In the previous chapter, the Robust TLNE for passive networks is explained. However, realistic power systems, especially large power systems, include generators and other active elements. Thus, it is necessary to extend the Robust TLNE approach to active networks. Moreover, a three-phase multi-port example system with an active external system will be modeled by Robust TLNE.

3.1 External System with Active Elements

Since external system is LTI, the contribution of active elements in the external system to the transients is only limited to power frequency. This gives the idea that we only need to consider the external system in power frequency and construct Norton equivalent current sources for active elements at external system input ports, as shown in Fig. 3.1. The Norton equivalent current sources are found by either using an analytical method [21, 22], or by measuring short-circuit current.

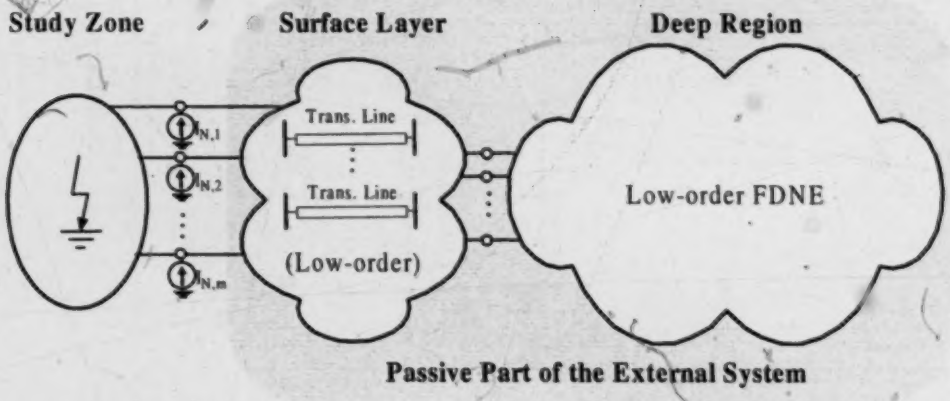


Figure 3.1: Robust TLNE model for external system with active elements

3.1.1 The Analytical Method

In power system transient studies, a generator can be treated as an ideal voltage source in series with an impedance. In an active external system, we denote the buses that have generators connected with subscript of G, interfacing buses to the study zone with subscript of A and all other buses with subscript of B. Then nodal equations are [21,22]

$$\begin{bmatrix} \mathbf{I}_A \\ \mathbf{I}_G \\ \mathbf{I}_B = 0 \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{AA} & \mathbf{Y}_{AG} & \mathbf{Y}_{AB} \\ \mathbf{Y}_{GA} & \mathbf{Y}_{GG} & \mathbf{Y}_{GB} \\ \mathbf{Y}_{BA} & \mathbf{Y}_{BG} & \mathbf{Y}_{BB} \end{bmatrix} \begin{bmatrix} \mathbf{V}_A \\ \mathbf{V}_G \\ \mathbf{V}_B \end{bmatrix} \tag{3.1}$$

The third row of (3.1) gives

$$\mathbf{V}_B = -\mathbf{Y}_{BB}^{-1}(\mathbf{Y}_{BA}\mathbf{V}_A + \mathbf{Y}_{BG}\mathbf{V}_G). \tag{3.2}$$

Substituting into first two rows of (3.1), we have

$$\begin{bmatrix} \mathbf{I}_A \\ \mathbf{I}_G \end{bmatrix} = \begin{bmatrix} \mathbf{Y}'_{AA} & \mathbf{Y}'_{AG} \\ \mathbf{Y}'_{GA} & \mathbf{Y}'_{GG} \end{bmatrix} \begin{bmatrix} \mathbf{V}_A \\ \mathbf{V}_G \end{bmatrix} \tag{3.3}$$

where

$$\begin{aligned} \mathbf{Y}'_{AA} &= \mathbf{Y}_{AA} - \mathbf{Y}_{AB}\mathbf{Y}_{BB}^{-1}\mathbf{Y}_{BA} \\ \mathbf{Y}'_{AG} &= \mathbf{Y}_{AG} - \mathbf{Y}_{AB}\mathbf{Y}_{BB}^{-1}\mathbf{Y}_{BG} \\ \mathbf{Y}'_{GA} &= \mathbf{Y}_{GA} - \mathbf{Y}_{GB}\mathbf{Y}_{BB}^{-1}\mathbf{Y}_{BA} \\ \mathbf{Y}'_{GG} &= \mathbf{Y}_{GG} - \mathbf{Y}_{AB}\mathbf{Y}_{BB}^{-1}\mathbf{Y}_{BG} \end{aligned} \tag{3.4}$$

The generator model is represented as

$$\mathbf{I}_G = \mathbf{Y}_G(\mathbf{E}_G - \mathbf{V}_G) \tag{3.5}$$

Substituting into (3.3), we obtain Norton equivalent

$$\mathbf{I}_A = \mathbf{Y}_N \mathbf{V}_A + \mathbf{I}_N \tag{3.6}$$

where

$$\begin{aligned} \mathbf{Y}_N &= \mathbf{Y}'_{AA} - \mathbf{Y}'_{AG}(\mathbf{Y}'_{GG} + \mathbf{Y}_G)^{-1}\mathbf{Y}'_{GA} \\ \mathbf{I}_N &= \mathbf{Y}'_{AG}(\mathbf{Y}'_{GG} + \mathbf{Y}_G)^{-1}\mathbf{Y}_G\mathbf{E}_G \end{aligned} \tag{3.7}$$

Thus, vector $\mathbf{I}_N$ is the Norton equivalent current sources at external system input ports; $\mathbf{Y}_N$ is the passive part of external system input admittance matrix at power frequency.

3.1.2 Measuring Short-Circuit Current

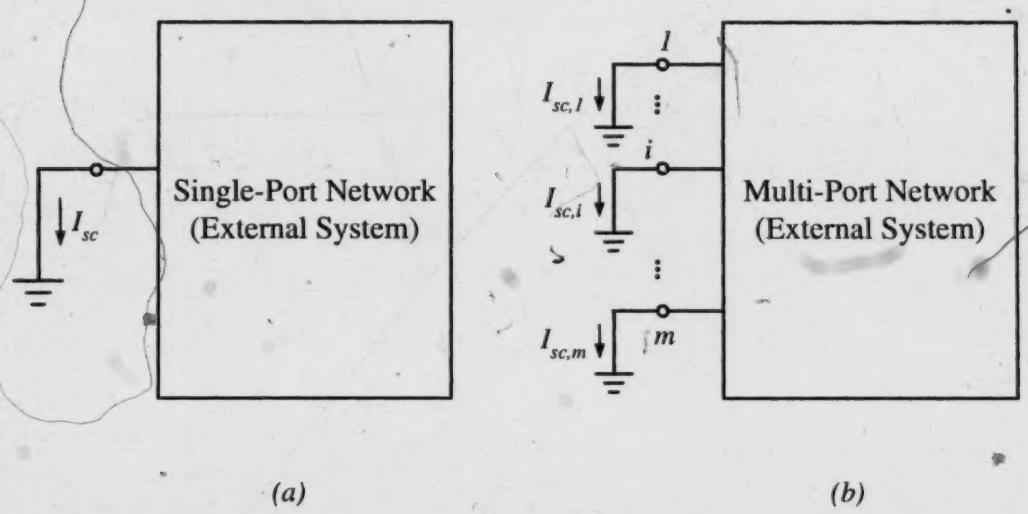


Figure 3.2: Obtaining Norton equivalent current sources for external system

Since the analytical method requires the reconstruction of external system admittance matrix, when the external system is very big, the procedure is error prone. Measuring short-circuit current is more reliable since the same circuits and data files as the ones for simulation in EMTP are used. In single-port external systems, the Norton equivalent current source is found by measuring short-circuit current at that port, shown in Fig. 3.2(a), where

$$I_N = I_{sc} \tag{3.8}$$

In multi-port external system case, shown in Fig. 3.2(b), the Norton equivalent current sources are obtained by measuring the short-circuit current at all ports at the same time.

In an m-port system, we have

$$\mathbf{I}_N = \mathbf{I}_{sc} \tag{3.9}$$

where $\mathbf{I}_N = [\ I_{N,1}\ \ I_{N,2}\ \ \cdots\ \ I_{N,m}\ ]^T$, $\mathbf{I}_{sc} = [\ I_{sc,1}\ \ I_{sc,2}\ \ \cdots\ \ I_{sc,m}\ ]^T$; $\mathbf{I}_N$ and $\mathbf{I}_{sc}$ correspond to the quantities in Figures 3.1 and 3.2(b), respectively.

The passive part of the external system is obtained by traditional method of eliminating all active elements, *i.e.*, voltage sources are considered short-circuit and current sources are considered open-circuit. The Robust TLNE model discussed in the previous chapter is applied to the passive part. Thus the procedures for building the Robust TLNE model for external system of active networks are obtained. Now, based on above explanation, it is necessary to illustrate the procedures to obtain Robust TLNE model, as shown in Fig. 3.3.

3.2 Robust TLNE for Three-Phase Multi-Port Systems

In three-phase systems, greatly attributed to transmission lines, mutual couplings exist. Therefore, some specific issues associated with three-phase systems requires to be discussed. In transient analysis, a three-phase system is commonly decoupled into three separate systems. Such decoupling is done by transformation between phase domain and modal domain. Since the external system is balanced and transposed, Clarke's transformation is more appropriate [45]. In addition, an alternative fitting routine for transmission line parameters is employed in multi-port systems with a strong constraint on passivity.

3.2.1 Clarke's Transformation

In a balanced and transposed power system, by Clarke's transformation, a three-phase system is decoupled into three separate single-phase systems called *modes*. The transformation matrix

$$\mathbf{T} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & \sqrt{2} & 0 \\ 1 & -1/\sqrt{2} & \sqrt{3}/\sqrt{2} \\ 1 & -1/\sqrt{2} & -\sqrt{3}/\sqrt{2} \end{bmatrix} \tag{3.10}$$

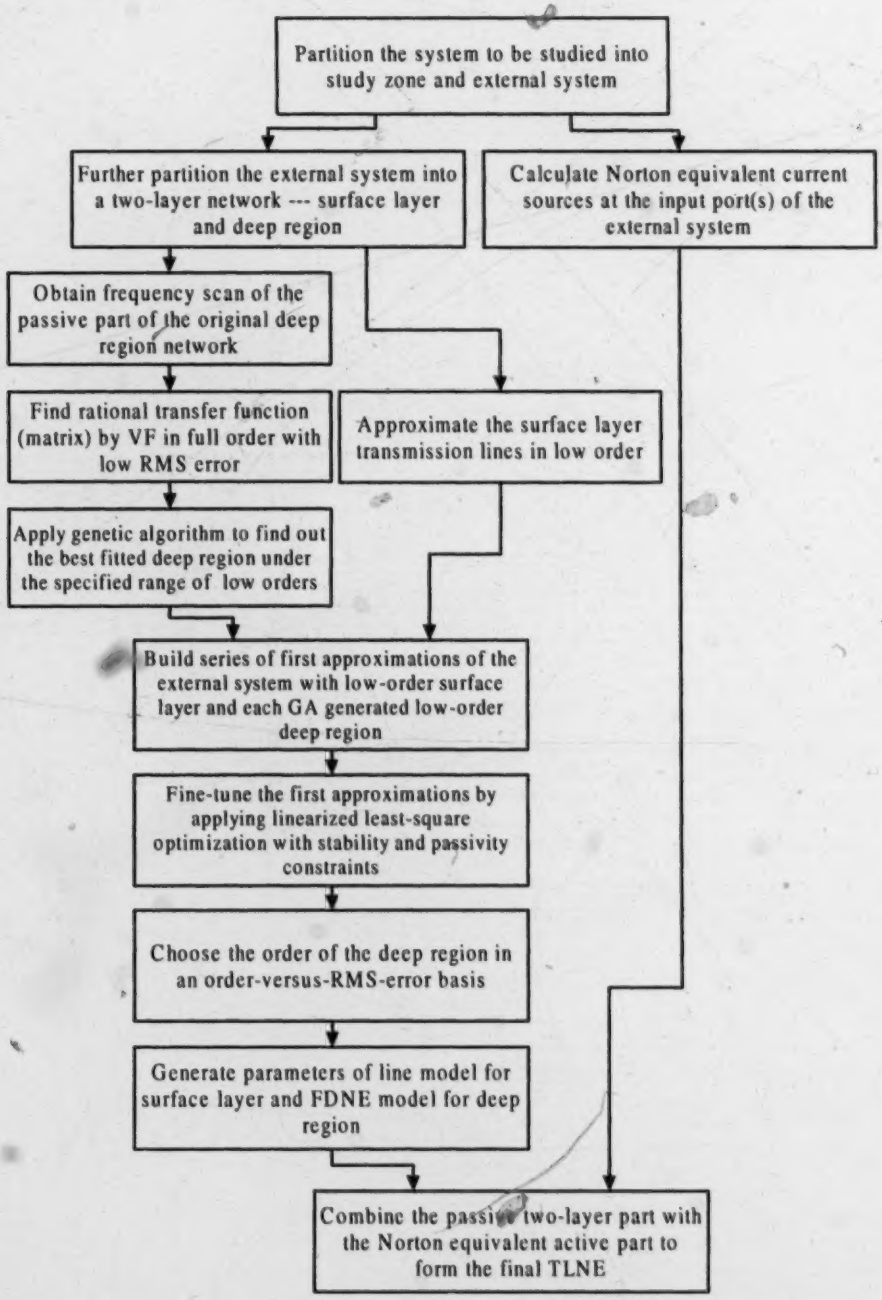

Partition the system to be studied into study zone and external system
Further partition the external system into a two-layer network --- surface layer and deep region
Calculate Norton equivalent current sources at the input port(s) of the external system
Obtain frequency scan of the passive part of the original deep region network
Find rational transfer function (matrix) by VF in full order with low RMS error
Approximate the surface layer transmission lines in low order
Apply genetic algorithm to find out the best fitted deep region under the specified range of low orders
Build series of first approximations of the external system with low-order surface layer and each GA generated low-order deep region
Fine-tune the first approximations by applying linearized least-square optimization with stability and passivity constraints
Choose the order of the deep region in an order-versus-RMS-error basis
Generate parameters of line model for surface layer and FDNE model for deep region
Combine the passive two-layer part with the Norton equivalent active part to form the final TLNE

Figure 3.3: Flowchart for obtaining Robust TLNE model for generic external systems

defines Clarke's transformation [45] as

$$\mathbf{i}_{mode} = \mathbf{T}^{-1}\mathbf{i}_{phase} \tag{3.11a}$$

$$\mathbf{v}_{mode} = \mathbf{T}^{-1}\mathbf{v}_{phase} \tag{3.11b}$$

$$\mathbf{Y}_{mode} = \mathbf{T}^{-1}\mathbf{Y}_{phase}\mathbf{T} \tag{3.11c}$$

where

$$\mathbf{i}_{phase} = [\ i_A \quad i_B \quad i_C\ ]^T \tag{3.12a}$$

$$\mathbf{v}_{phase} = [\ v_A \quad v_B \quad v_C\ ]^T \tag{3.12b}$$

$$\mathbf{Y}_{phase} = \begin{bmatrix} Y_s & Y_m & Y_m \\ Y_m & Y_s & Y_m \\ Y_m & Y_m & Y_s \end{bmatrix} \tag{3.12c}$$

$$\mathbf{i}_{mode} = [\ i_0 \quad i_\alpha \quad i_\beta\ ]^T \tag{3.13a}$$

$$\mathbf{v}_{mode} = [\ v_0 \quad v_\alpha \quad v_\beta\ ]^T \tag{3.13b}$$

$$\mathbf{Y}_{mode} = \begin{bmatrix} Y_0 & 0 & 0 \\ 0 & Y_\alpha & 0 \\ 0 & 0 & Y_\beta \end{bmatrix} \tag{3.13c}$$

and

$$Y_0 = Y_s + 2Y_m,\ Y_\alpha = Y_\beta = Y_s - Y_m \tag{3.14}$$

$\mathbf{T}$ is orthogonal, which means

$$\mathbf{T}^{-1} = \mathbf{T}^T = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ \sqrt{2} & -1/\sqrt{2} & -1/\sqrt{2} \\ 0 & \sqrt{3}/\sqrt{2} & -\sqrt{3}/\sqrt{2} \end{bmatrix} \tag{3.15}$$

Thus, a balanced m-port three-phase system defined by equation

$$\begin{bmatrix} \mathbf{i}_{P,1} \\ \mathbf{i}_{P,2} \\ \vdots \\ \mathbf{i}_{P,m} \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{P,11} & \mathbf{Y}_{P,12} & \cdots & \mathbf{Y}_{P,1m} \\ \mathbf{Y}_{P,21} & \mathbf{Y}_{P,22} & \cdots & \mathbf{Y}_{P,2m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Y}_{P,m1} & \mathbf{Y}_{P,m2} & \cdots & \mathbf{Y}_{P,mm} \end{bmatrix} \begin{bmatrix} \mathbf{v}_{P,1} \\ \mathbf{v}_{P,2} \\ \vdots \\ \mathbf{v}_{P,m} \end{bmatrix} \tag{3.16}$$

is transformed to modal domain using Clarke's transformation as

$$\begin{bmatrix} \mathbf{i}_{M,1} \\ \mathbf{i}_{M,2} \\ \vdots \\ \mathbf{i}_{M,m} \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{M,11} & \mathbf{Y}_{M,12} & \cdots & \mathbf{Y}_{M,1m} \\ \mathbf{Y}_{M,21} & \mathbf{Y}_{M,22} & \cdots & \mathbf{Y}_{M,2m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Y}_{M,m1} & \mathbf{Y}_{M,m2} & \cdots & \mathbf{Y}_{M,mm} \end{bmatrix} \begin{bmatrix} \mathbf{v}_{M,1} \\ \mathbf{v}_{M,2} \\ \vdots \\ \mathbf{v}_{M,m} \end{bmatrix} \tag{3.17}$$

where subscript $_P$ stands for phase-domain quantities, $\mathbf{i}_{P,i}$, $\mathbf{v}_{P,i}$ $(1 \leq i \leq m)$ are phase-domain ABC current and voltage vectors, and $\mathbf{Y}_{P,ij}$ $(1 \leq i,j \leq m)$ are symmetrical admittance matrices of phase domain as (3.12c); subscript $_M$ denotes modal-domain quantities, $\mathbf{i}_{M,i}$, $\mathbf{v}_{M,i}$ $(1 \leq i \leq m)$ are modal-domain $0\alpha\beta$ current and voltage vectors, and $\mathbf{Y}_{M,ij}$ $(1 \leq i,j \leq m)$ are diagonal admittance matrices of modal domain as (3.13c). Relationships between modal-domain and phase-domain quantities are

$$\mathbf{i}_{M,i} = \mathbf{T}^{-1}\mathbf{i}_{P,i} \tag{3.18a}$$

$$\mathbf{v}_{M,i} = \mathbf{T}^{-1}\mathbf{v}_{P,i} \tag{3.18b}$$

$$\mathbf{Y}_{M,ij} = \mathbf{T}^{-1}\mathbf{Y}_{P,ij}\mathbf{T} \tag{3.18c}$$

After Clarke's transformation, there are only two separate networks required solving during simulation, since α network is the same as β network in modal domain due to balanced systems. The α and β modes are equal and denoted as *aerial modes* or *sky modes*; the 0/zero mode is also called *ground mode*.

During EMTP simulation of three-phase systems, system equations are solved in phase domain to obtain nodal voltages. Then history current terms are updated in modal domain and transformed back to phase domain.

3.2.2 Nonlinear Fitting Method for Transmission Line Parameters

Example 1 in Section 2.6 is a single-port network, in which positive-real criterion is not a strong constraint during the final fine-tuning via CLLSQ optimization. However, in multi-port systems, our experience shows that system margin associated with positive-real constraint is very limited, *i.e.*, parameter change is significantly confined by positive-real criterion. In such cases, although surface layer and deep region compensate each other, they may not be able to compensate larger deviations, which is mostly caused by low-order fitting of surface layer, since deep regions generated by GAs are very accurate. Thus, RMS-error% during fitting transmission line $Z_c(\omega)$ and $P(\omega)$ must be kept very low, *e.g.*, below 1%, with more accuracy stressed at lower frequencies and power frequency. Nonetheless, within this RMS-error% range, Bode's asymptotic fitting technique implemented by most of EMTP packages often produces high-order rational functions. In Robust TLNE to make transmission line models more suitable for real-time simulation, a nonlinear fitting technique due to Fernandes *et al.* [42,

43] is used. This method is an iterative nonlinear least-square optimization procedure based on Levenberg-Marquardt approach.

In multi-phase systems, after Clarke's transformation, commonly the aerial mode exhibits more resonant peaks in frequency response, whereas the frequency response of the ground mode appears to be smoother. Our experience also shows that after VF the system with smoother frequency response normally has more margin for optimization than the one with a lot of resonant peaks. Thus, during fitting for surface layer transmission line parameters, RMS-error% allowance for ground mode is set to less than 1% and for aerial mode is only 0.5% max.

3.3 Example 2

The 240kV system used here [23] is a modification of a standard system for transient stability studies [50]. Fig. 3.4(a) shows the system single-line diagram. The transient phenomena to be analyzed are balanced capacitor switching event at Bus 15 and three-phase to ground fault at Bus 16. In both cases, voltage and current are measured at Bus 16. System elements' parameters are shown in Appendix B.1. All transmission towers are assumed to have the same configurations and profiles. To simplify the original model construction, all parameters of generators and transformers were converted to 240kV base.

3.3.1 Generation of the Robust TLNE

The system is first partitioned into a study zone and an external system, as shown in Fig. 3.4(a). Fig. 3.4(b) shows the passive part of the external system. According to guidelines in Section 1.2 with engineering judgement, in the passive part of the external system, transmission line TL1, TL2 and TL3 are considered for the surface layer and the remaining system of a two-port network comprises the deep region.

Frequency scan (discussed in Section 2.1.2) of the original deep region network provided the phase-domain admittance matrix $\mathbf{Y}_{deep,P}(\omega)$. Clarke's transformation further decoupled $\mathbf{Y}_{deep,P}(\omega)$ to ground mode $\mathbf{Y}_{deep,0}(\omega)$ and aerial mode $\mathbf{Y}_{deep,\alpha}(\omega)$ in modal domain. Shown in Figures 3.6 through 3.8, due to ground return, the ground mode tends to be smoother. Full-order VF generates a 60th-order rational function matrix

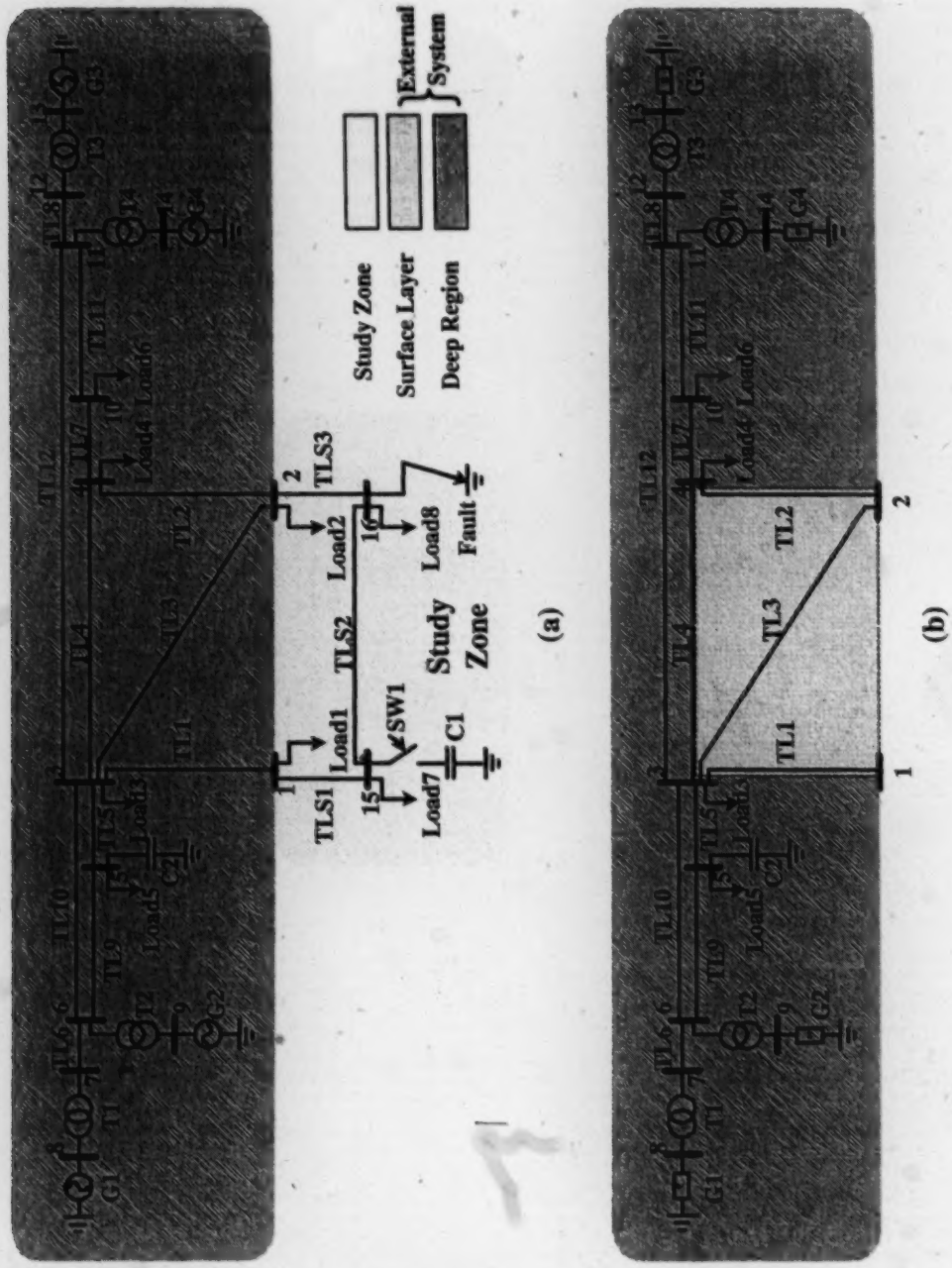


Figure 3.4: Example 2 system. (a) Example 2 system diagram and its partitioning. (b) Example 2 passive part of the external system.

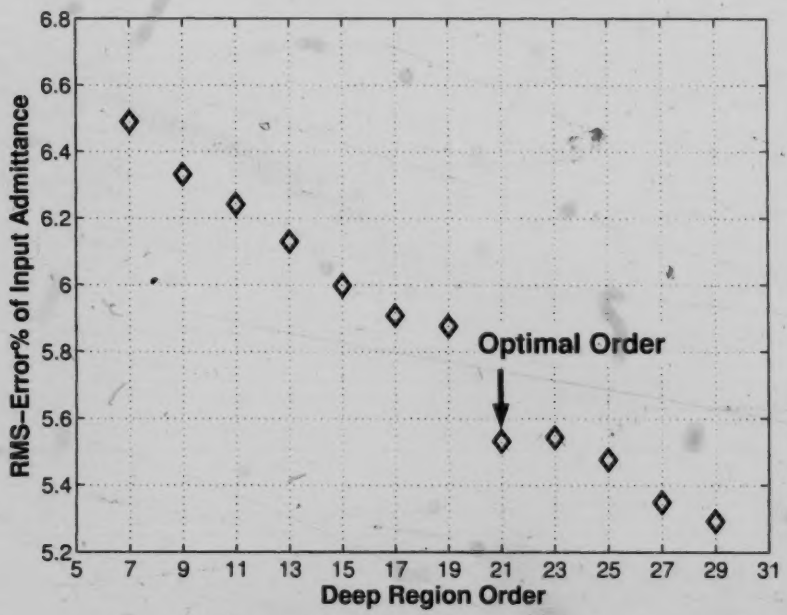


Figure 3.5: Example 2 aerial mode RMS-error% of input admittance *v.s.* deep region order

with 0.43% RMS-error. On the contrary, the aerial mode exhibits a lot of resonant peaks, as shown in Figures 3.12 through 3.14. Fitting such frequency response, VF generates a 220th-order rational function matrix of 4 real poles and 108 complex conjugate pole pairs with 4.38% RMS-error. Following the rules of Robust TLNE, the 4 partial fractions with real poles are selected for deep region and all partial fractions with complex pairs are to be processed by GAs.

Due to the multi-port nature and passivity constraints of the system, in fitting the surface layer parameters $Z_c(\omega)$ and $P(\omega)$, non-linear fitting technique [42, 43] is used. Since the transients to be analyzed are balanced, only aerial mode is considered for both surface layer transmission lines and deep region networks. The RMS-error% of input admittance *v.s.* deep region order is shown in Fig. 3.5, from which, order 21 was found to be the optimal order for the deep region with low RMS-error of 5.533%. Figures 3.12 through 3.14 show the deep region frequency response generated by GAs and after CLLSQ optimization. Figures 3.9 through 3.11 show the input admittances of ex-

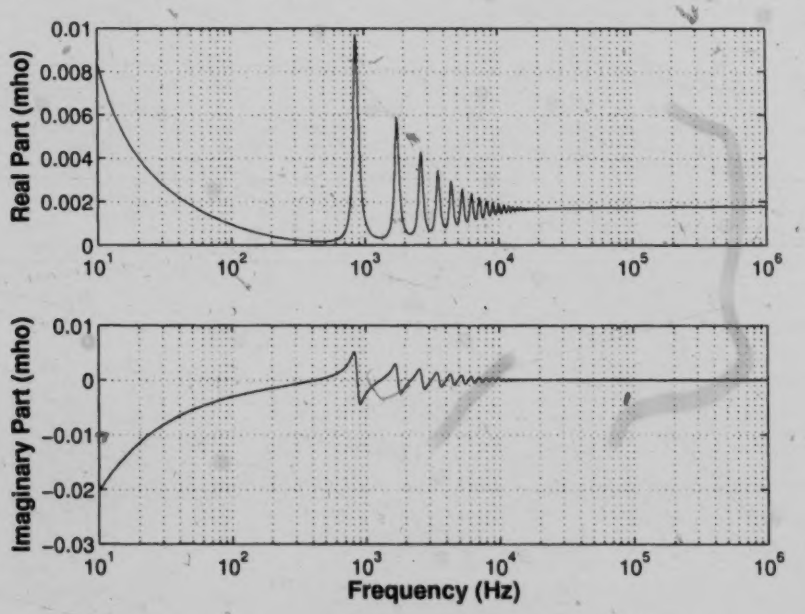


Figure 3.6: Example 2 ground mode input admittance $\mathbf{Y}_{input,0,11}$

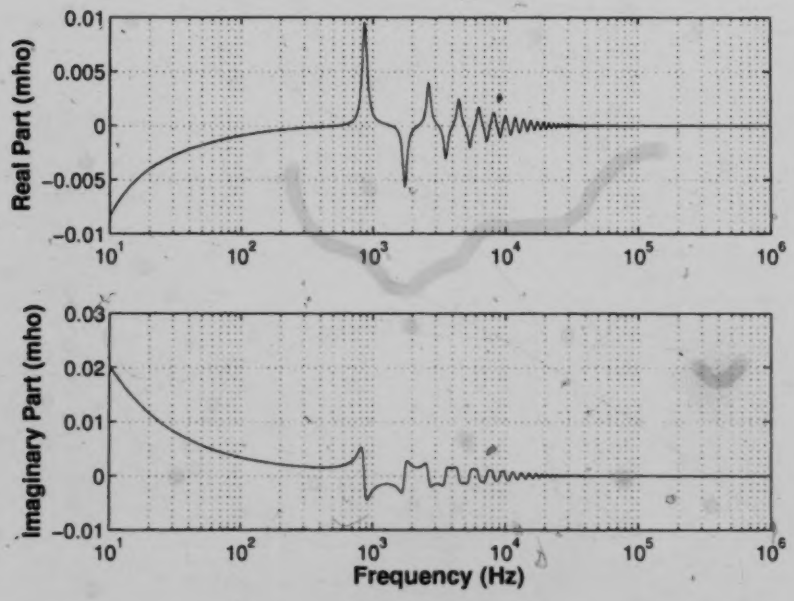


Figure 3.7: Example 2 ground mode input admittance $\mathbf{Y}_{input,0,12}$

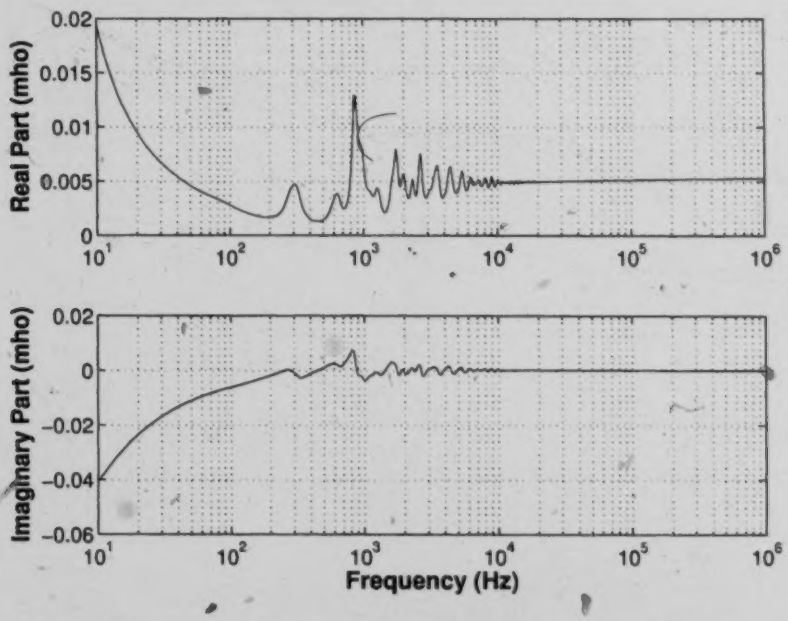


Figure 3.8: Example 2 ground mode input admittance $\mathbf{Y}_{input,0,22}$

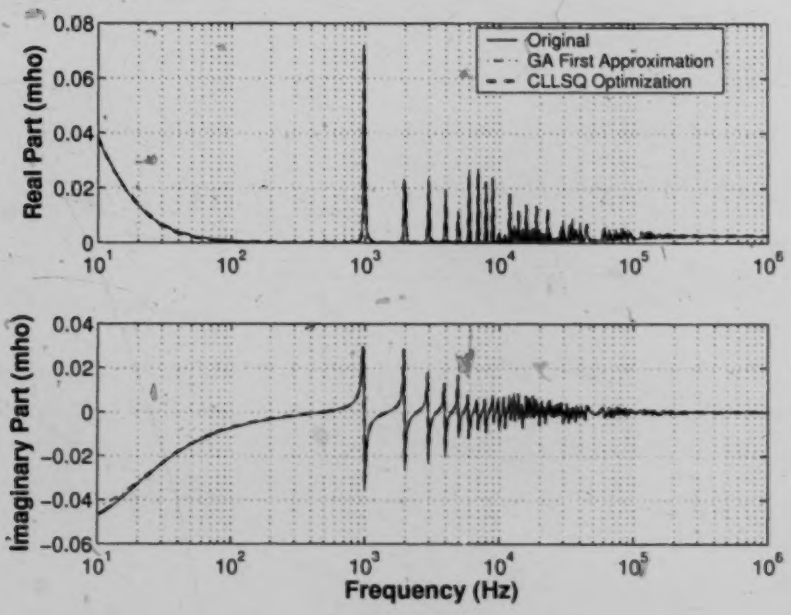


Figure 3.9: Example 2 aerial mode input admittance $\mathbf{Y}_{input,\alpha,11}$

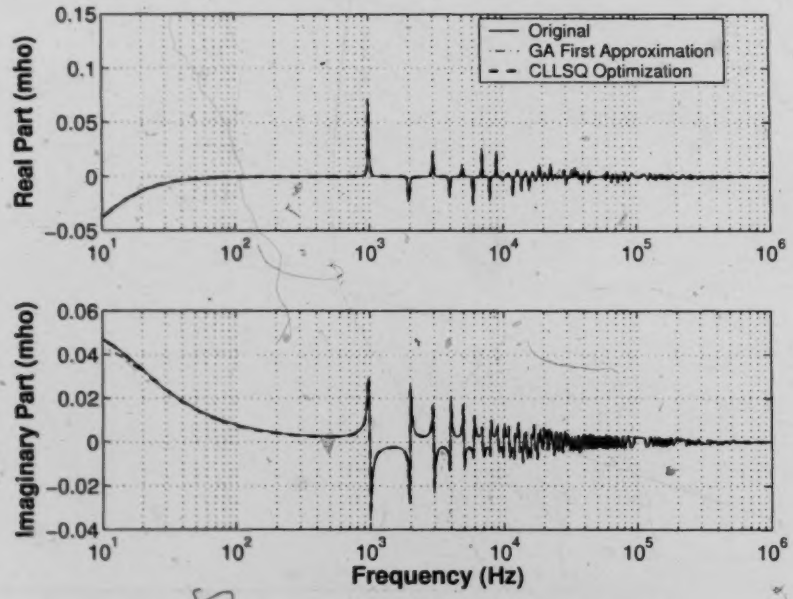


Figure 3.10: Example 2 aerial mode input admittance $\mathbf{Y}_{input,\alpha,12}$

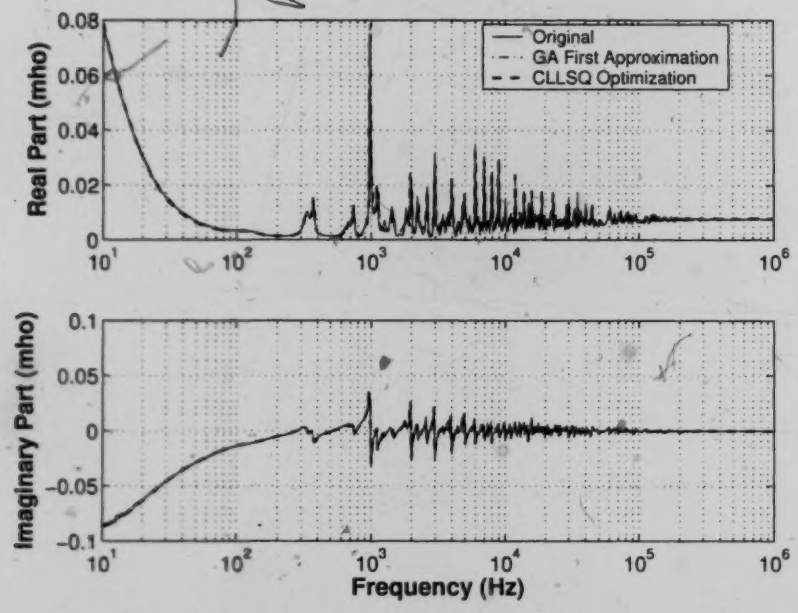


Figure 3.11: Example 2 aerial mode input admittance $\mathbf{Y}_{input,\alpha,22}$

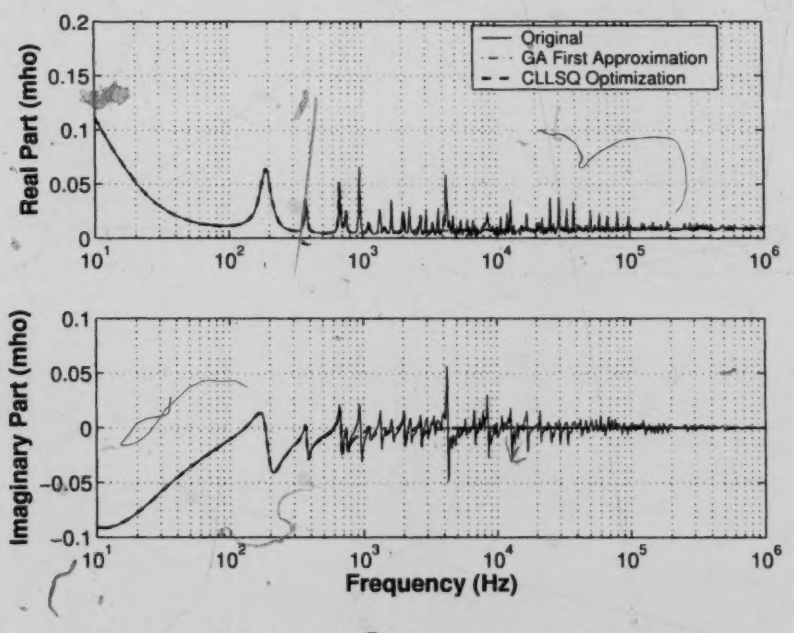


Figure 3.12: Example 2 aerial mode deep region admittance $\mathbf{Y}_{deep,\alpha,11}$

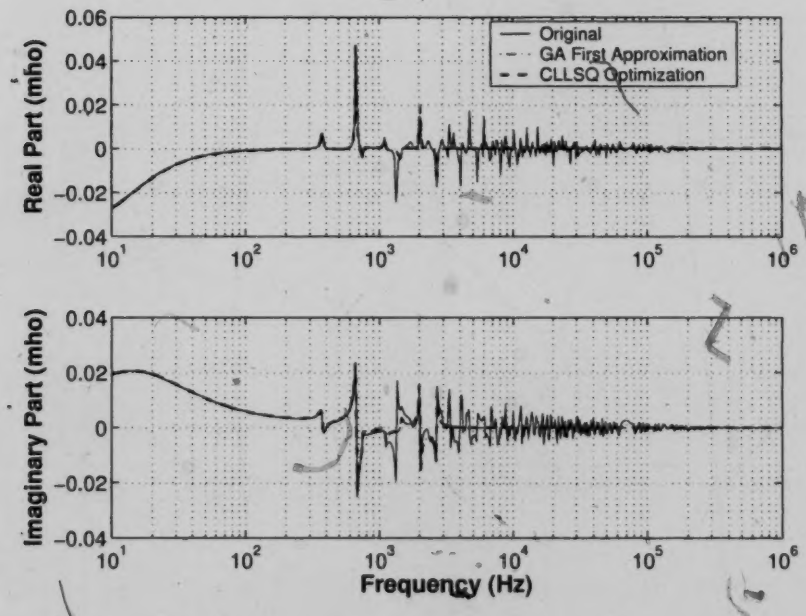


Figure 3.13: Example 2 aerial mode deep region admittance $\mathbf{Y}_{deep,\alpha,12}$

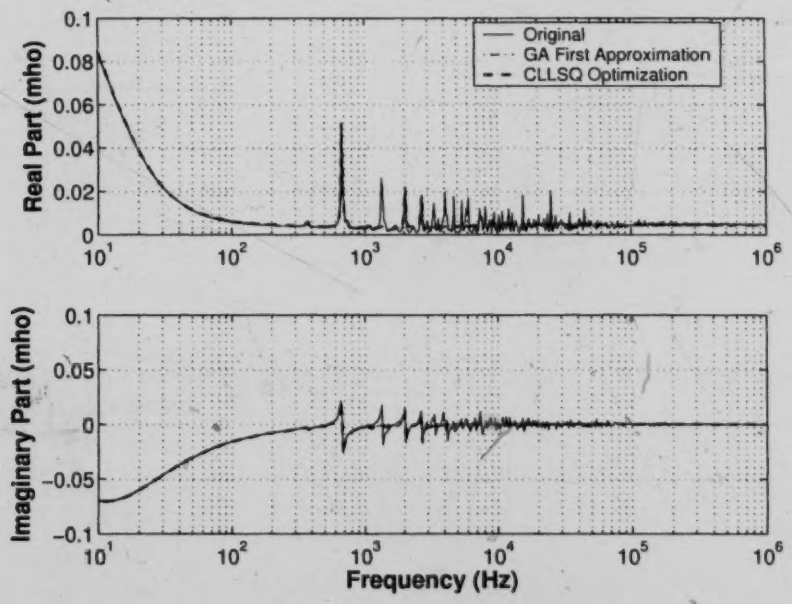


Figure 3.14: Example 2 aerial mode deep region admittance $\mathbf{Y}_{deep,\alpha,22}$

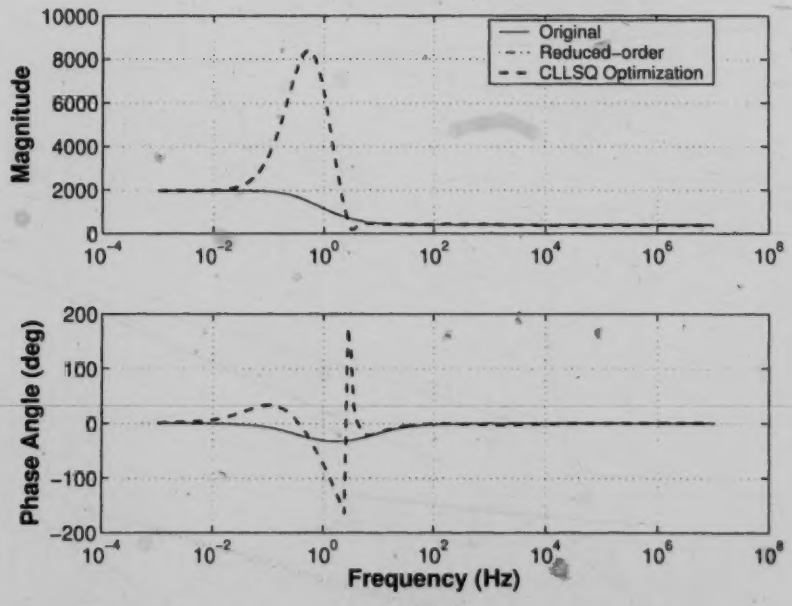


Figure 3.15: Example 2 TL1 aerial mode characteristic impedance

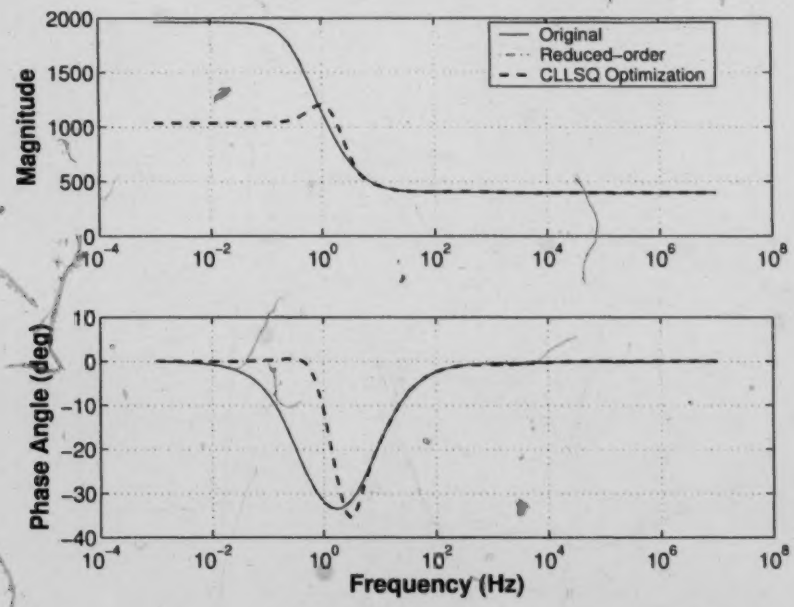


Figure 3.16: Example 2 TL2 aerial mode characteristic impedance

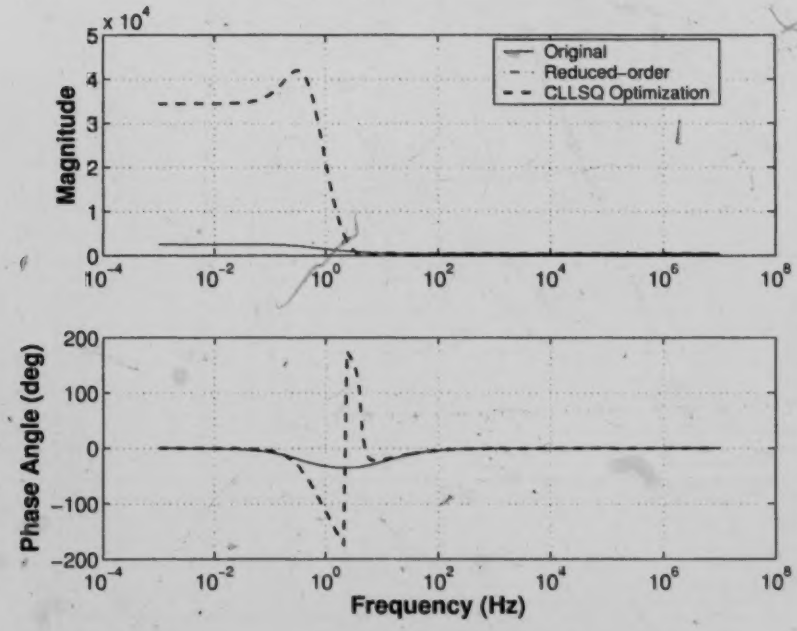


Figure 3.17: Example 2 TL3 aerial mode characteristic impedance

ternal system. It can be observed that the first approximations of input admittances are very close to original. Frequency responses of both the external system and deep region after CLLSQ optimization overlap those generated from GAs. It is shown that due to their relative insensitivity to the input admittance, some pronounced resonant peaks in deep region are not chosen by GAs. Figures 3.15 through 3.17 show characteristic impedance of surface layer transmissions TL1, TL2 and TL3 respectively. Further CLLSQ optimization mostly enhances the accuracy in the low frequency range, especially at DC, where the maximum RMS-error% is only 2.4%. Aerial mode transmission line parameters in surface layer are changed to compensate deviations in input admittance. Norton equivalent current sources for the external system are obtained through measuring short circuit current (Section 3.1.2) at input terminals. The phase-A phasors at each port are 1.648$\angle$-86.21°kA and 2.284$\angle$-93.66°kA, respectively. Converting to modal domain, the Norton equivalent current source phasor vectors $I_{eq,0\alpha\beta}$ are

$$[\ 0\angle 0° \quad 2.018\angle\text{-}86.21° \quad 2.018\angle\text{-}176.21°\ ]^T \text{ kA}$$

and

$$[\ 0\angle 0° \quad 2.797\angle\text{-}93.66° \quad 2.797\angle\text{-}183.66°\ ]^T \text{ kA}$$

at each port, respectively.

3.3.2 Transient Simulations

Transient events	Total time	Full model	Robust TLNE	160th FDNE
C1 switching	0.15s	1.034s	0.082s	1.024s
Balanced fault	0.20s	1.072s	0.117s	1.064s

Table 3.1: Example 2 computational time comparison at time step size 20μs

Fig. 3.18 shows phase-A voltage and current transients at Bus 15 where capacitor C1 is switched at 0.05s. Three-phase transients of this event are shown Fig. 3.19. Figures 3.20 and 3.21 show phase-A and three-phase fault current and voltage transients also at Bus 15, when balanced three-phase to ground fault is induced at Bus 16 with a 2Ω fault resistance per phase. The fault occurs at 0.05s. All transients are verified via ATP

with time step size 20μs. Detailed agreement between the full model of the system and the the robust TLNE model is observed. Computational time is a major saving in robust TLNE. Table 3.1 shows computational time comparison among full model, Robust TLNE model and FDNE model on a Pentium IV 1.6GHz computer. The FDNE model with 160th order has 6.859% RMS-error (higher than the TLNE model) and does not demonstrate great savings on computational time due to its high order. As seen from Table 3.1, the Robust TLNE model is 9 to 12 times faster than the full model, which makes it highly attractive for real-time digital simulation. Application of the existing method [21,22] to obtain TLNE of the same order produces 10.53% RMS-error, which is higher than the new approach. The ATP data files for the full model and Robust TLNE model are shown in Appendix B.3 and B.4, respectively.

3.4 Summary

In this chapter, the Robust TLNE model is extended to active networks and multi-phase multi-port systems. Procedures to obtain Norton equivalent current sources and interaction between phase domain and modal domain via Clarke's transformation are explained in detail. An accurate alternative technique in fitting transmission line parameters by nonlinear Levenberg-Marquardt method is also discussed. A three-phase multi-port example is modeled by the Robust TLNE to further verify the validity and stability of proposed approach.

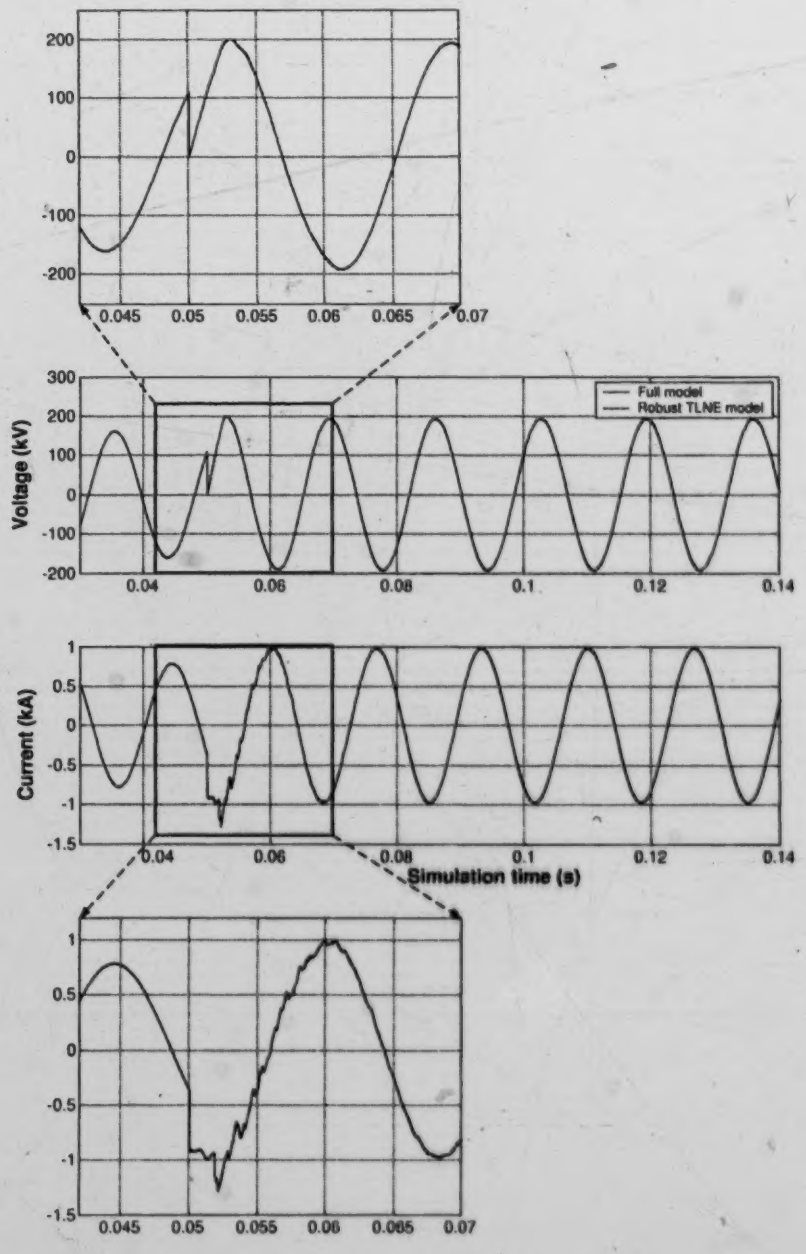


Figure 3.18: Example 2 $C1$ switching transient simulation and comparison (phase A)

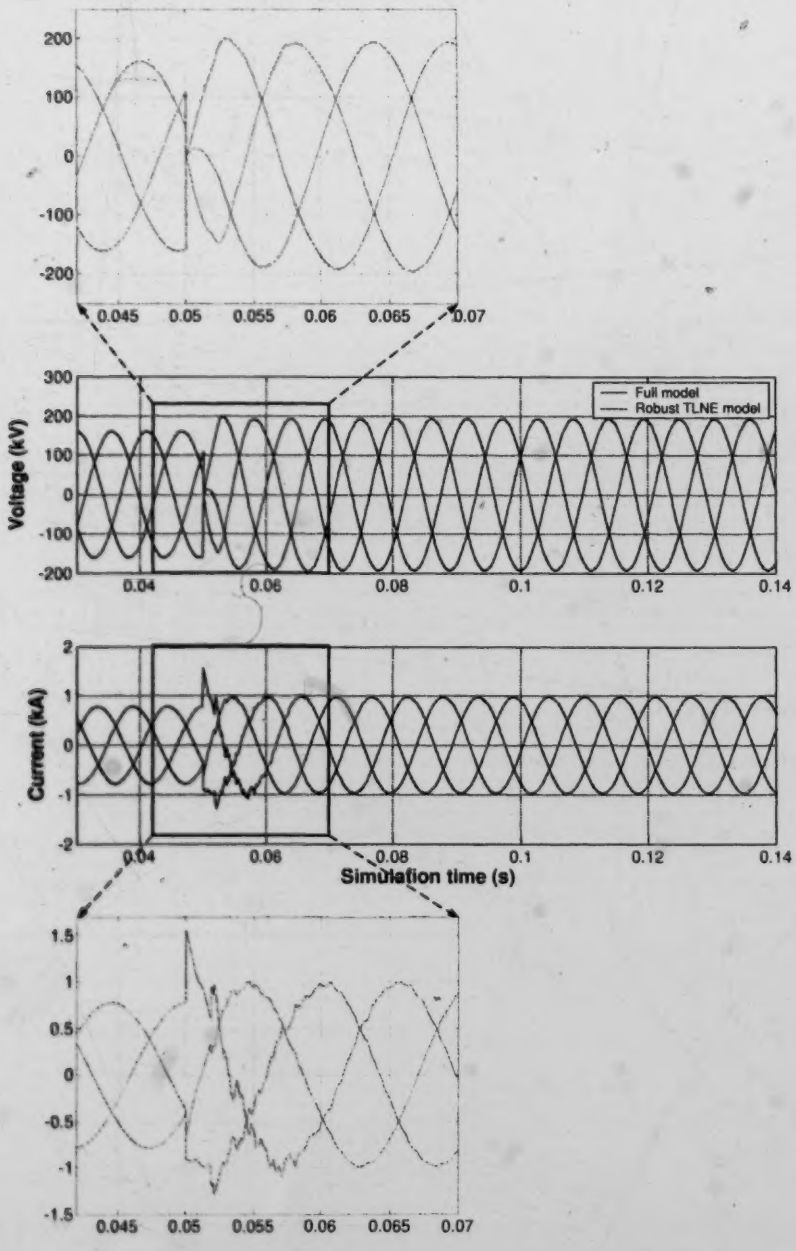


Figure 3.19: Example 2 $C1$ switching transient simulation and comparison (three phase)

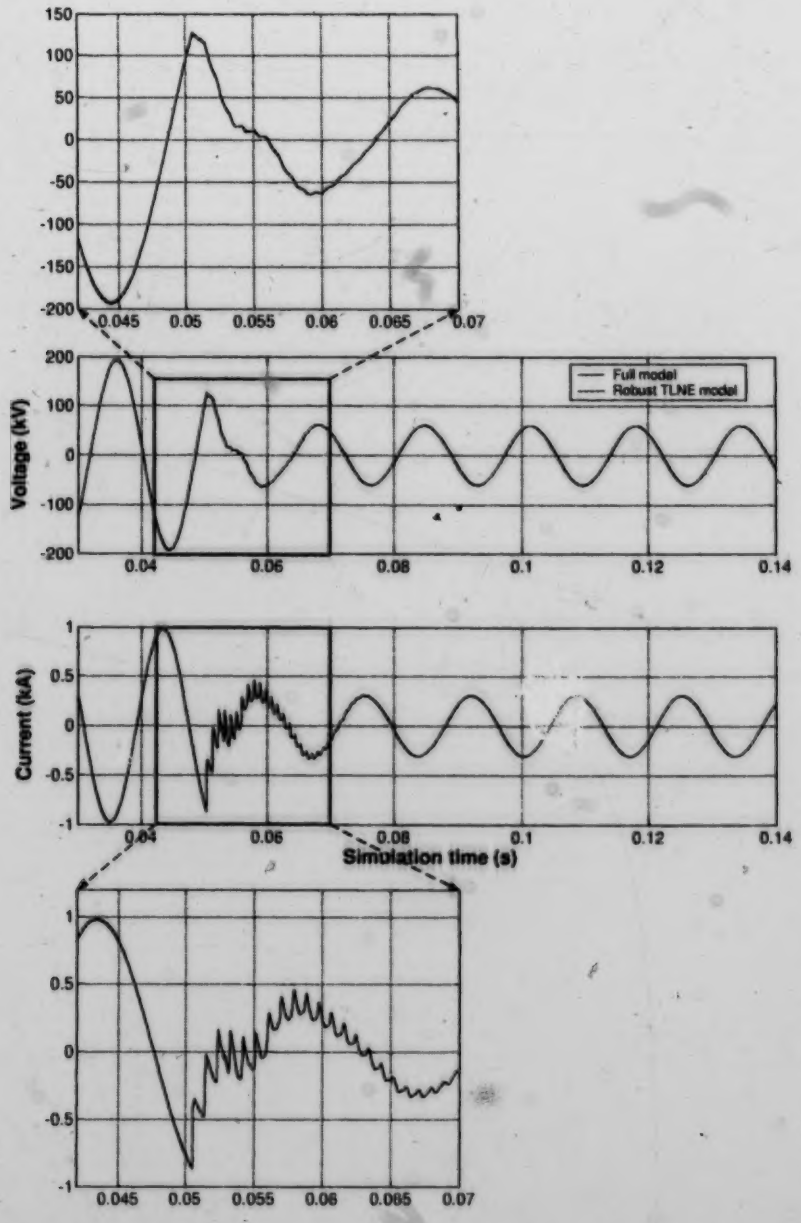


Figure 3.20: Example 2 three-phase to ground fault transient simulation and comparison (phase *A*)

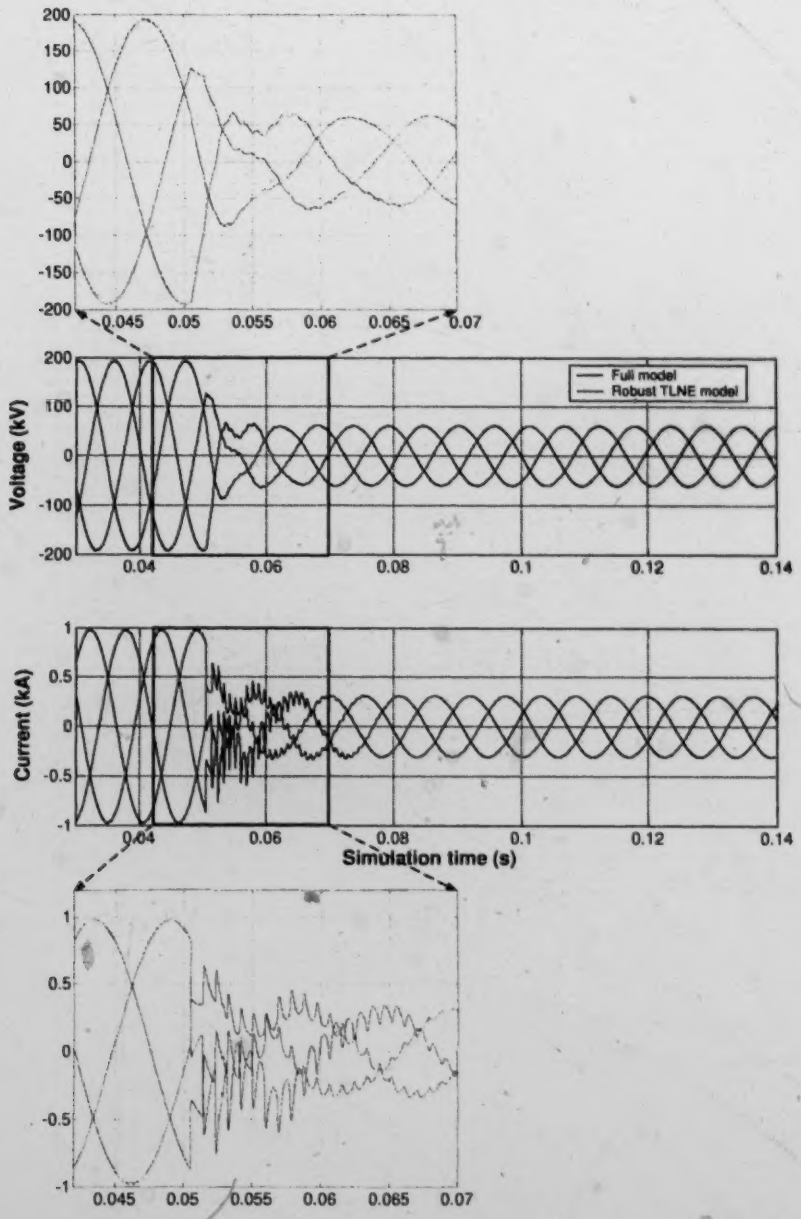


Figure 3.21: Example 2 three-phase to ground fault transient simulation and comparison (three phase)

4

Case Study — the Alberta Interconnected Electric System

With procedures for modeling Robust TLNE expanded and two examples explained in previous chapters, the accuracy and computational efficiency of the Robust TLNE model make it well-suited for real-time simulation of large power systems. In this chapter, a realistic large scale power system — the Alberta Interconnected Electric System (AIES) is to be modeled by a Robust TLNE at Bus 524 Genesee. Balanced three-phase to ground fault transients are to be examined and compared between the original full model and Robust TLNE model.

4.1 The Alberta Interconnected Electric System

The AIES, shown in Fig. 4.1, is a trans-province transmission and generation network with over 8,000MW generation capacities. As of 2003, it consists of over 17,000km of transmission lines and more than 400 substations ranging in voltage level from 69kV to 500kV. Two inter-province connections, which are HVDC transmission to SaskPower (Saskatchewan Province) and 500kV transmission lines to BC Hydro (British Columbia Province), connect AIES to the North American electric power grid. The AIES keeps growing along with Alberta's economy.

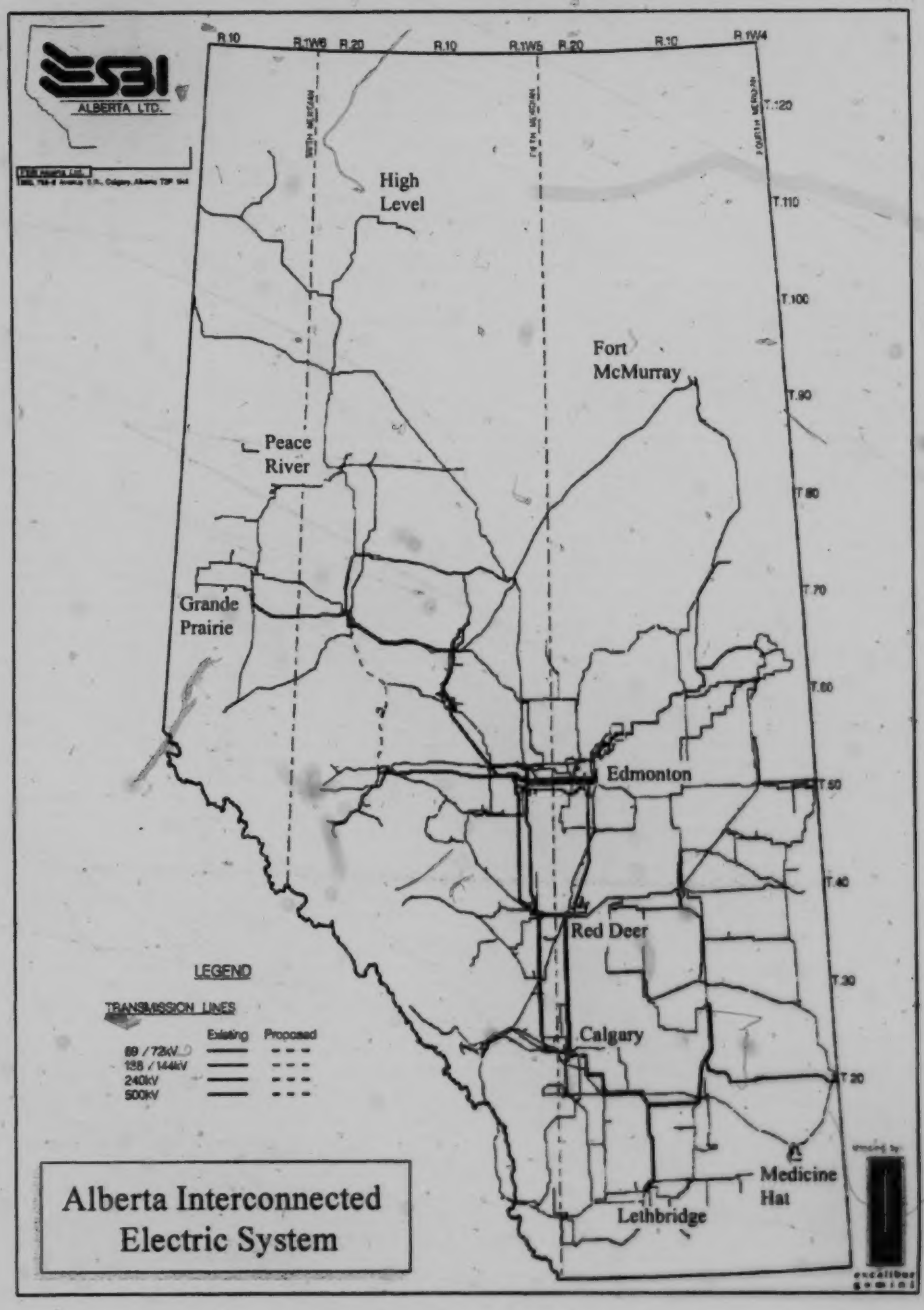


Figure 4.1: Map of AIES (Courtesy of AESO)

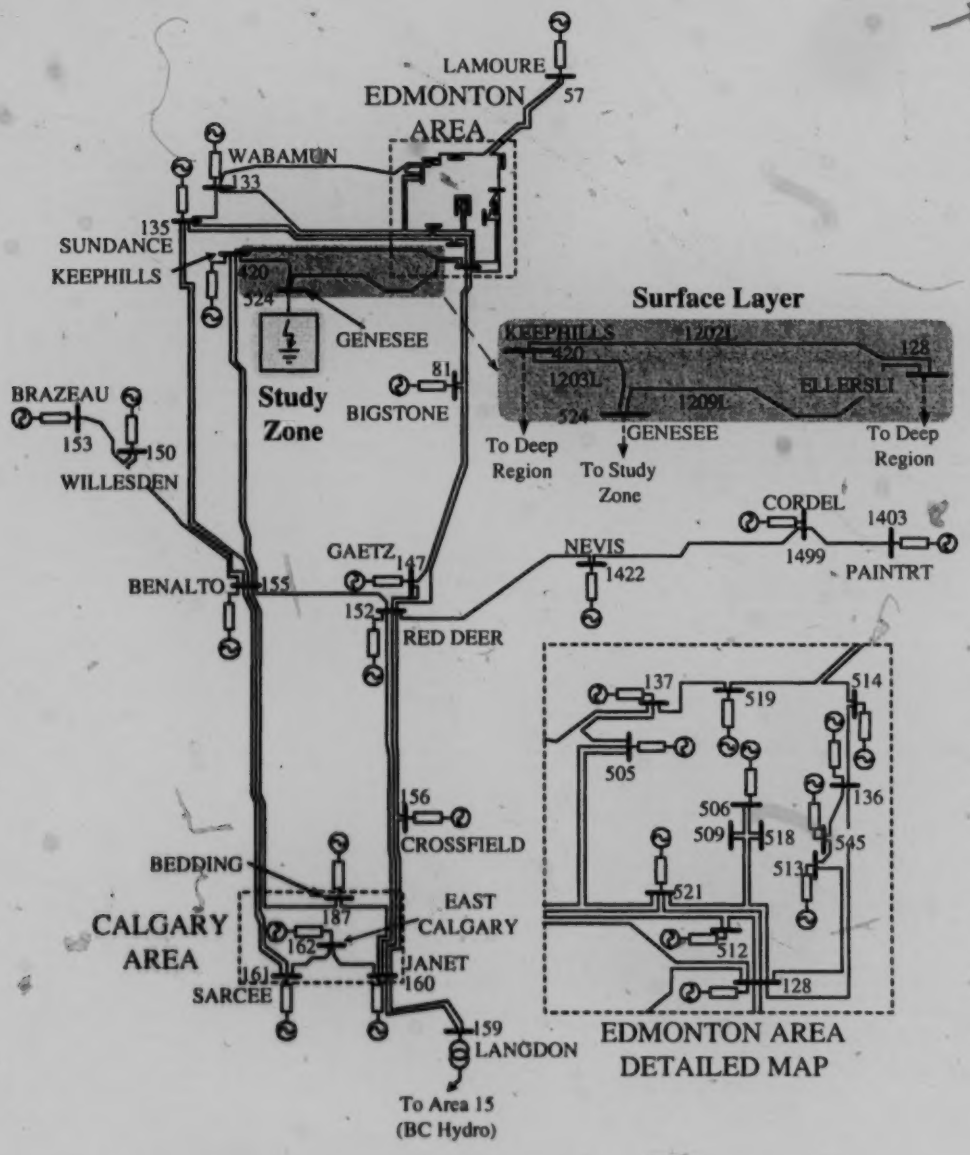


Figure 4.2: AIES Area 50 Backbone Single-Line Diagram

The system to be studied is 2003 Spring medium load operational case. In PSS/E, this case study of AIES has over 1600 buses with unique names, which are classified into approximately fifty numbered areas. Each area is also designated a name, *e.g.*, "Backbone" referring to Area 50, "WSCC" referring to Area 15, "Edmonton" referring

to Area 60, "Calgary" referring to Area 6, *etc.*. SaskPower and BC Hydro connections are modeled as Thevenin equivalents. The Area 50 Backbone is an essential part of AIES, most buses of which are 240kV voltage level, as shown in Fig. 4.2. Mainly, the Area 50 includes one large ring network, which is from Edmonton Area to Calgary Area.

The transients to be analyzed are at Bus 524 Genesee, which belongs to Area 50 Backbone. It connects a number of generation stations together to the power grid of AIES. Due to the excessive complexities in modeling of AIES, network reduction is carried out while retaining the model's high accuracy.

4.2 Transmission Administrator System Model of AIES

The Transmission Administrator System Model or TASMo of AIES, is a model of all elements and facilities, that affect electricity flows through the Alberta Interconnected grid. Realized in Microsoft® Access, the TASMo database[1] provides detailed information of the following system elements and facilities:

- Transmission lines and line segments of each transmission line,
- Substations,
- Transformers,
- Machines including generators and motors,
- Shunts including capacitors and reactors,
- Loads.

A relational database is a collection of tables among which relationships exist. The TASMo database is constructed in the same way. All elements and facilities in AIES appear relationally in the corresponding tables. For example, table `TOPO_BUSSES` provides bus name, voltage level, ownership, PSS/E area code, facility name the bus resides as well as typical voltage and angle. The information provided in the TASMo database is suitable for both power flow and dynamic studies.

[1]The latest development of TASMo is called TASMo2 based on Oracle® Database.

4.3 Modeling and Network Reduction of AIES

In order to reduce the difficulties in modeling such a big power system in EMTP, the AIES is first equivalenced to only include Area 50 240kV Backbone, since transients occur in this area. In PSS/E, the appropriate activity for equivalencing AIES is `eeqv`, which allows to eliminate buses on the area and bus basis without affecting power flow. All buses in other areas except Area 50 are equivalenced. Appendix. C.2 shows detailed procedures to obtain equivalents for other areas excluding Area 50. Fig. 4.2 also illustrates the equivalents for outer areas with respect to Area 50. The equivalents obtained in PSS/E are all in per unit format for branches and MW/MVar format for loads and shunts. However, ATP requires values in Ω, mH, or μF. Thus, it is necessary to convert those equivalents into ATP readable style data. The following provides the methodology for the conversion:

- Equivalents appear in PSS/E with circuit number 99. Therefore, retained elements in Area 50 and equivalents can be distinguished.
- The base MVA for AIES is 100MVA. At 240kV voltage level, the base value for resistance, reactance or impedance is

  $$R_{base,240} = X_{base,240} = Z_{base,240} = 576\Omega$$

  and the base value for conductance, susceptance or admittance is

  $$G_{base,240} = B_{base,240} = Y_{base,240} = 0.001736\mho$$

  Then actual values for equivalent branches in Ω are obtained from PSS/E through above equations and are input into ATPDraw.
- Since loads are assumed to have constant real and reactive power, a constant current source is used for each load. The current source phasor is calculated by

  $$I\angle\theta_I = \frac{\sqrt{(2)}\,\vec{S}^*}{240\sqrt{3}\,\vec{V}^*}$$

  where I is the current source magnitude, θ_I is the current source phase shift, $\vec{S}$ is the complex power absorbed by the load, $\vec{V}$ is the voltage phasor at the bus calculated in load flow, and superscript * denotes complex conjugate.

- Shunts are represented as RLC circuits and obtained from

$$Z_{shunt} = \frac{V^2}{\overrightarrow{S}^*}$$

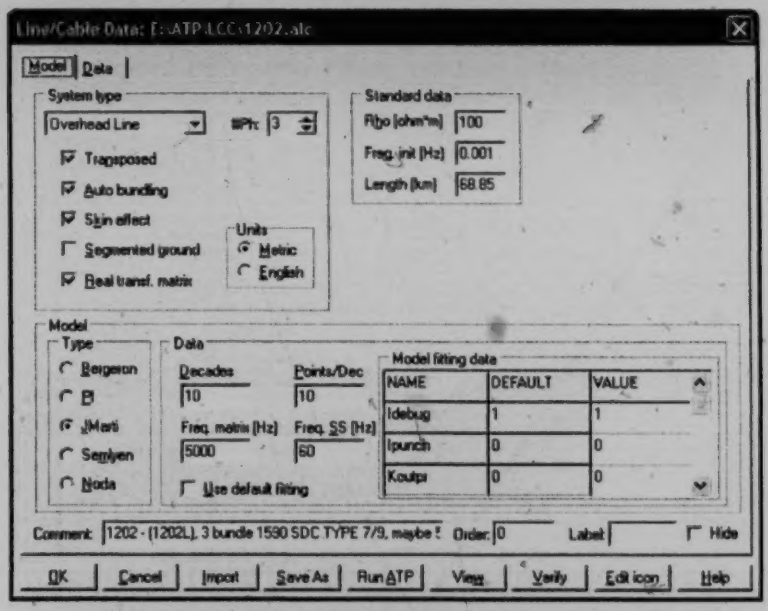
where V is base voltage level at the bus which is 240kV, and $\overrightarrow{S}$ is the complex power absorbed by the shunt.

- For convenience purpose, in ATP, the inductance is expressed as a reactance in Ω, while the capacitance is expressed as a susceptance in $\mu\mho$. Thus `XOPT` and `COPT` variables in ATP are both set to be 60.

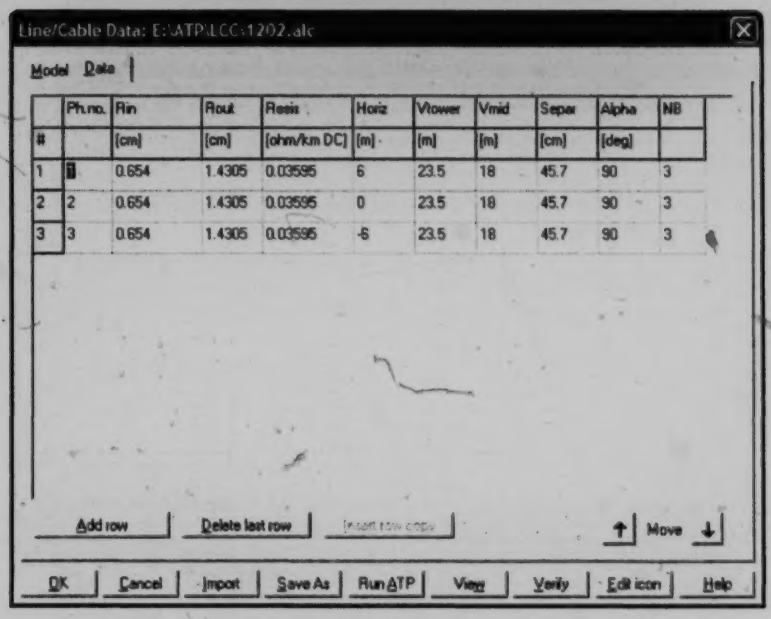
More accurate analysis of Area 50 requires all transmissions lines in the system modeled with frequency-dependence. This also enables the consideration of mutual couplings between some double circuit lines sharing the same right-of-way. In ATP, such frequency-dependent lines are modeled based on conductor geometry, which is not available in power flow data provided in PSS/E. Nonetheless, the AIES TASMo database provides adequate information for building a detailed model of frequency-dependent lines for Area 50 in ATP. In TASMo database

- Table `STD_TOWERCOORDS` provides tower code and phase coordinations.
- Conductor type and characteristics are given in table `STD_CONDUCTORS`.
- The names and connections of transmission line segments can be found in table `TOPO_BRANCH_LINESEGS`.
- Table `TOPO_BRANCH_LINESEGS` supplies detailed information for each line segment, such as conductor name, number of bundles, bundle spacing, tower code, height of the tower and length of the line segment.

Therefore, frequency-dependent lines in AIES Area 50 can be built in ATPDraw, which is shown in Fig. 4.3. Conductor characteristics can be found in the appendix of [51]. With above explanation, by combining the peripheral equivalent circuits generated from PSS/E with detailed model provided from TASMo, the AIES Area 50 Backbone model in ATP for electromagnetic transient analysis is obtained. Fig. 4.4 shows a part of actual EMTP model built in ATPDraw. The full diagram is shown in Appendix C.1. The corresponding ATP data file is shown in Appendix C.3.1, which is the base file for transient studies in the following sections.



(a) The line model information



(b) The line data information

Figure 4.3: Building a frequency-dependent transmission line from TASMo database in ATPDraw

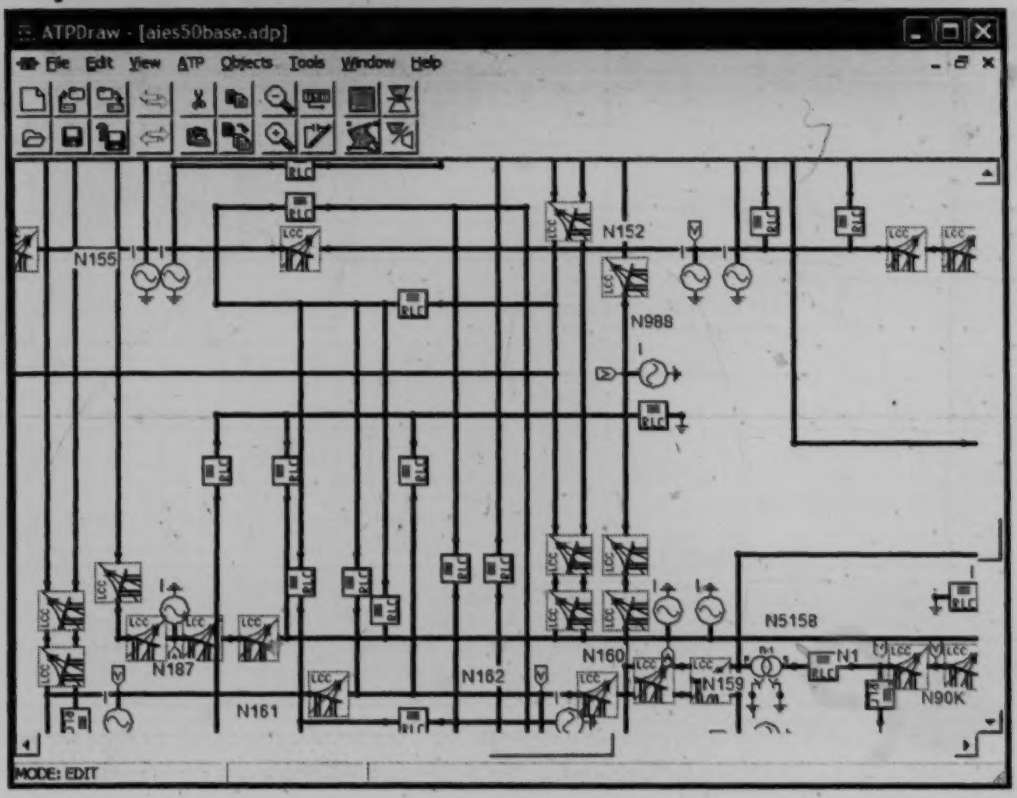


Figure 4.4: A part of AIES Area 50 diagram in ATPDraw

4.4 The Robust TLNE model for AIES

Since transients to be analyzed are around Bus 524 (Genesee), the generation stations are treated as a study zone and the rest of the network is considered as an external system. Furthermore, shown in Fig. 4.2, after applying TLNE model to the external system, transmission lines 1202L, 1203L, 1209L belong to the surface layer and the remaining of the external system forms the deep region. Shown in Fig. 4.6, due to ground return, the ground-mode input admittance $\mathbf{Y}_{input,0}(\omega)$ is very smooth. Full-order VF only generates a 36th-order FDNE model with 1.28% RMS-error. Therefore, the Robust TLNE model is only applied to aerial-mode admittance $\mathbf{Y}_{deep,a}(\omega)$, since very high order rational function is required to achieve low RMS-error% in VF due to large amount of resonant peaks, as shown in Fig. 4.7.

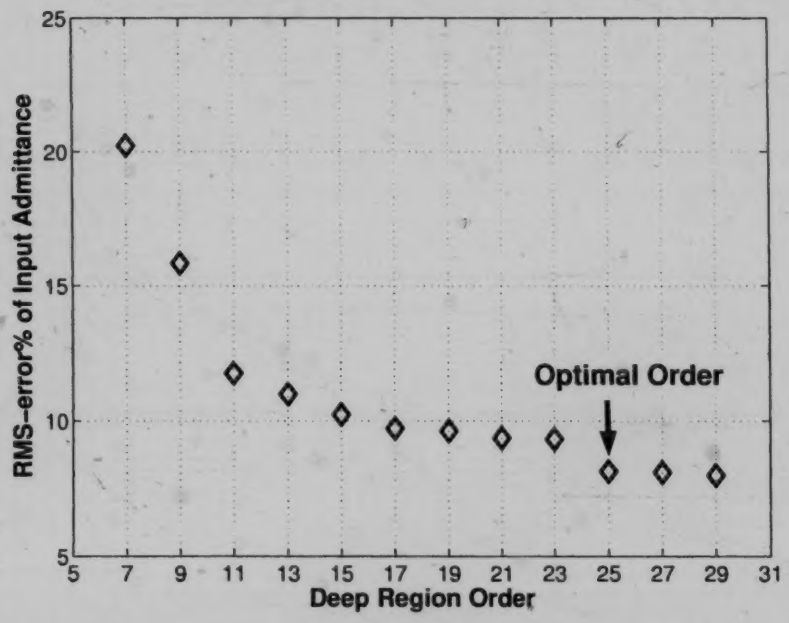


Figure 4.5: AIES Area 50 aerial mode RMS-error% of input admittance versus deep region order

The deep region is a two-port three-phase network. After removing all sources in the deep region, frequency scan of original deep region network gives phase-domain frequency response of deep region admittance matrix $\mathbf{Y}_{deep,P}(\omega)$. Clarke's transformation produces aerial mode $\mathbf{Y}_{deep,\alpha}(\omega)$ in modal domain. The aerial mode exhibits a lot of resonant peaks, as shown in Figures 4.8 through 4.10. In VF, the upper limit is 10kHz, since resonant peaks in higher frequency range are of less significant effects than those in lower frequency range, which is illustrated in both Example 1 and Example 2. With this method, VF generates a 240th-order rational function matrix of 2 real poles below 60Hz and 119 complex conjugate pole pairs with 1.93% RMS-error at discrete frequency points from 10Hz to 10kHz. In fitting of the whole frequency range from 10Hz to 1MHz, with the same level of RMS-error%, VF generates a 560th-order rational function matrix. Following the rules of Robust TLNE, the partial fractions with real poles are selected for deep region and all partial fractions of complex pairs are to be processed by GAs.

Due to the multi-port nature and complex frequency response of the system, in fitting the surface layer transmission line parameters $Z_c(\omega)$ and $P(\omega)$, non-linear fitting

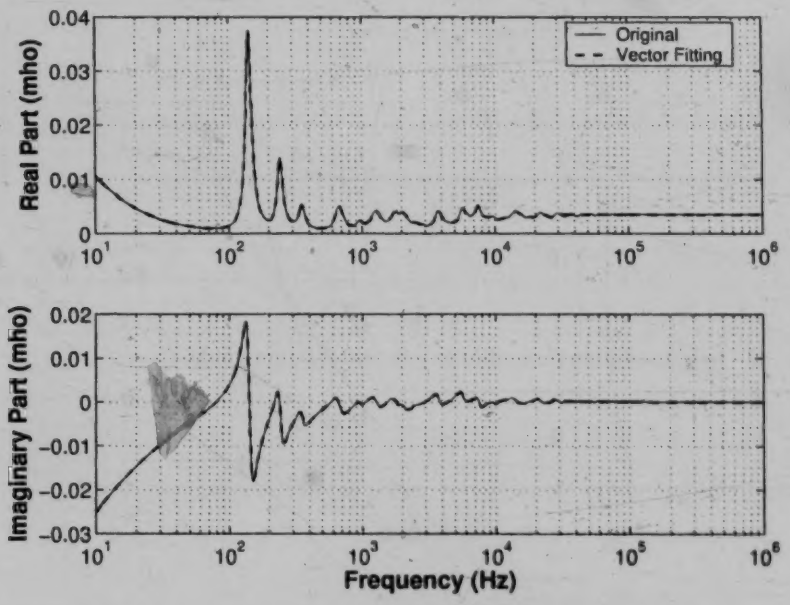


Figure 4.6: AIES Area 50 ground mode input admittance $\mathbf{Y}_{input,0}$

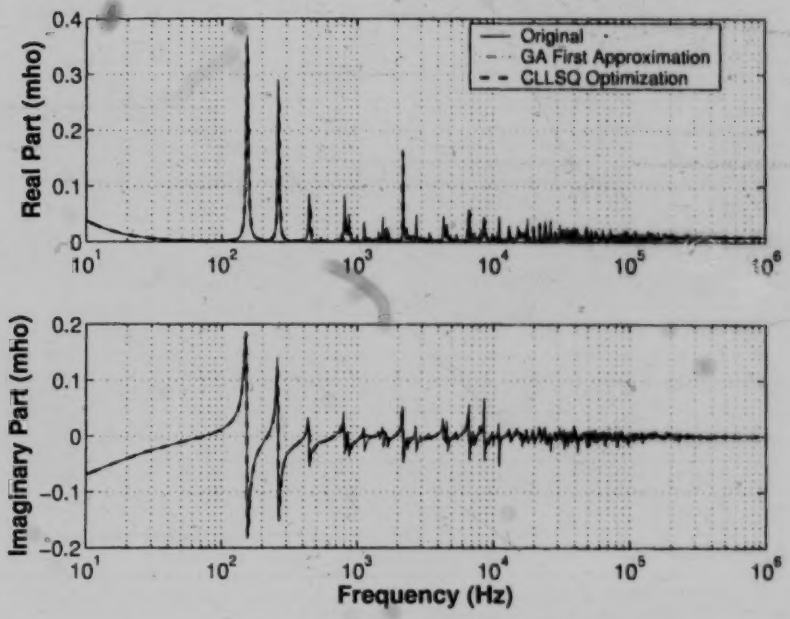


Figure 4.7: AIES Area 50 aerial mode input admittance $\mathbf{Y}_{input,a}$

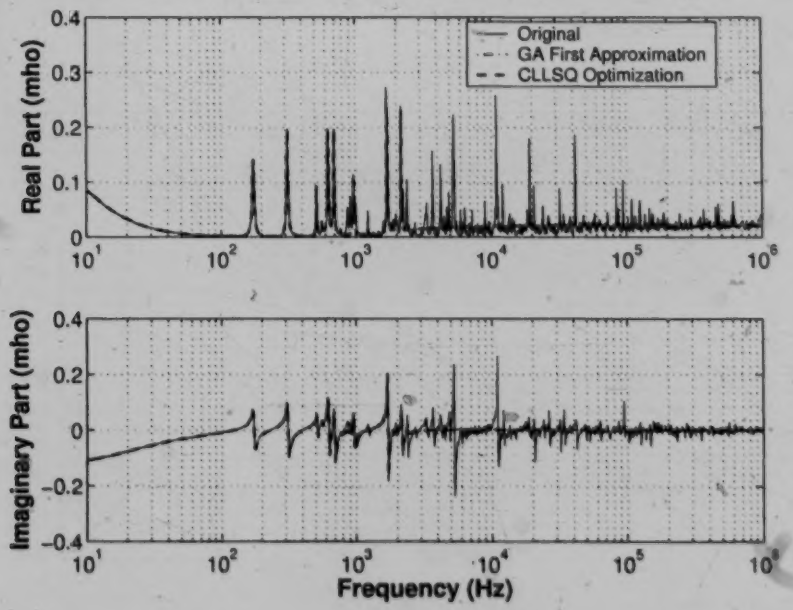


Figure 4.8: AIES Area 50 aerial mode deep region admittance $\mathbf{Y}_{deep,a,11}$

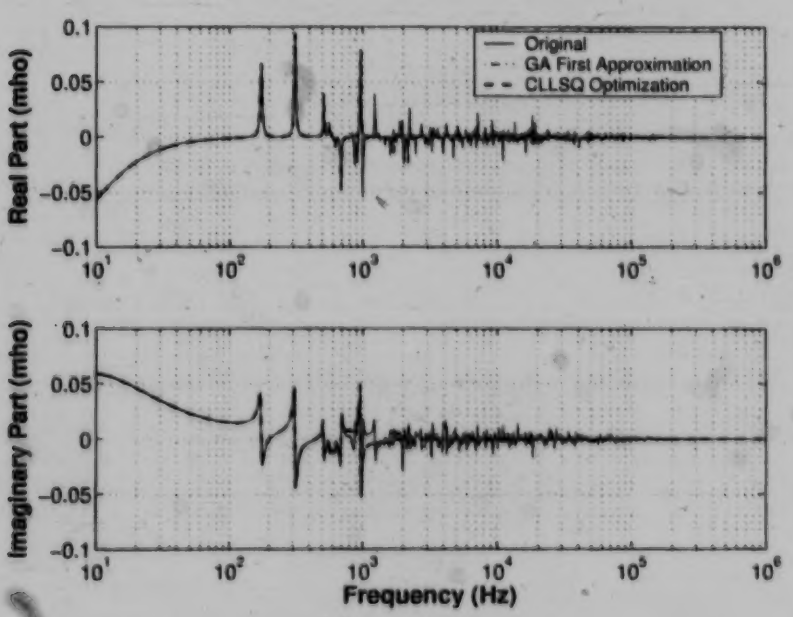


Figure 4.9: AIES Area 50 aerial mode deep region admittance $\mathbf{Y}_{deep,a,12}$

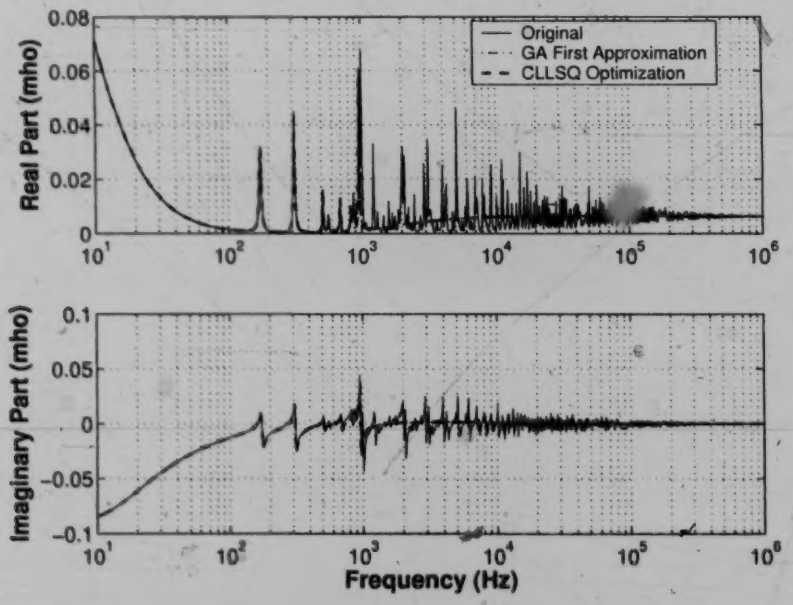


Figure 4.10: AIES Area 50 aerial mode deep region admittance $\mathbf{Y}_{deep,a,22}$

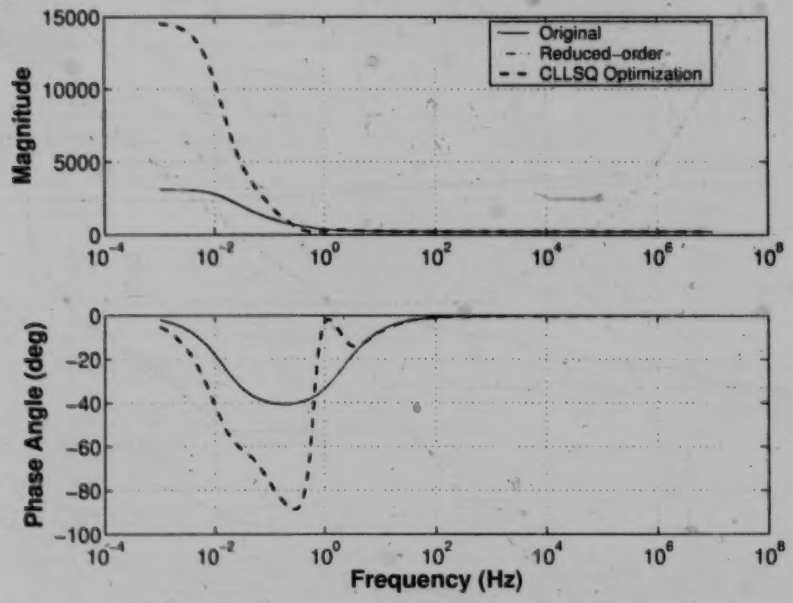


Figure 4.11: AIES Area 50 1202L aerial mode characteristic impedance

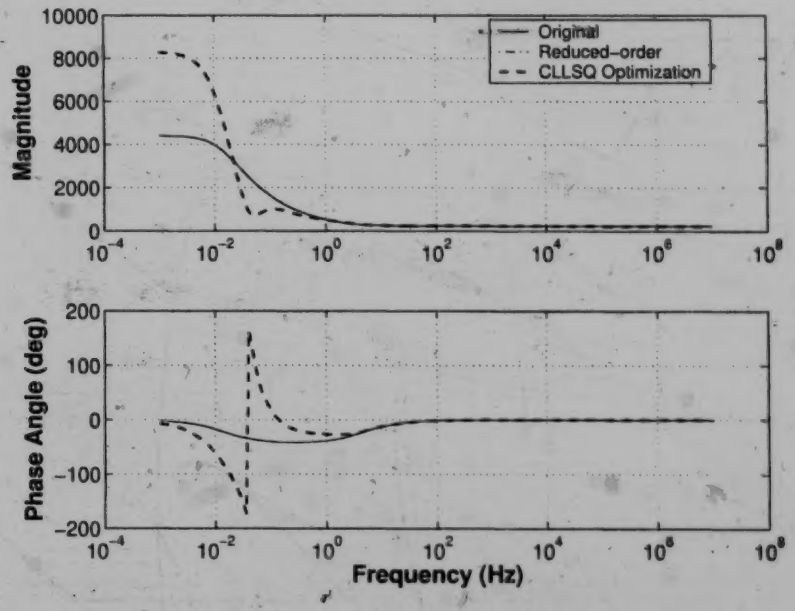


Figure 4.12: AIES Area 50 1203L aerial mode characteristic impedance

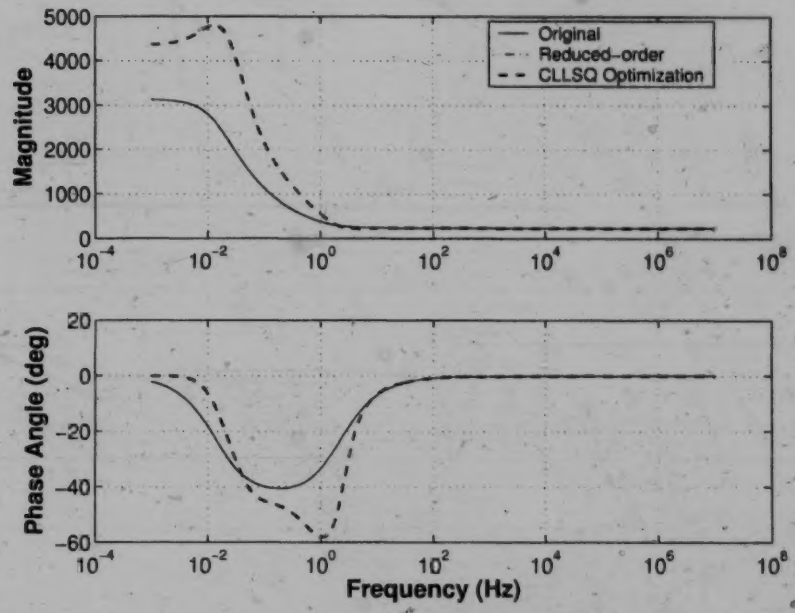


Figure 4.13: AIES Area 50 1209L aerial mode characteristic impedance

technique [42, 43] is used for aerial mode. Combining the surface layer with the results computed by GAs, CLLSQ is applied to find out the best suitable deep regions. The RMS-error% of input admittance versus deep region order is shown in Fig. 4.5 for aerial mode. The order of 25 was recognized as the optimal deep region order where with the higher orders of deep region, RMS-error% of input admittance does not decrease dramatically and is less than 10% (8.144%). Figures 4.8 through 4.10 show the aerial-mode deep region frequency response generated by GAs and after CLLSQ optimization. Fig. 4.7 shows the aerial-mode input admittance of external system. It can be observed that the first approximation of input admittance is very close to original. Frequency response of the external system input admittance after CLLSQ optimization overlaps that of GA first approximations. It is shown that due to their relative insensitivity to the input admittance, some pronounced resonant peaks in deep region are not chosen by GAs, which is salient feature of Robust TLNE. Further CLLSQ optimization mostly enhances the accuracy in the low frequency range, especially at DC, where the maximum RMS-error% is only 1.089%. This can be noticed from the frequency response of the surface layer characteristic impedance (shown in Figures 4.11 through 4.13), since in lower frequency range, it changes for the compensations on the frequency response at DC. The Norton equivalent current source for the external system is obtained through measuring short circuit current (Section 3.1.2) at input terminal, *i.e.*, Bus 524. The phase-*A* phasor at the input port is 1.881$\angle$-125.63° kA. Converting to modal domain, the Norton equivalent current source phasor vector $I_{eq,0\alpha\beta}$ is

$$\left[\begin{array}{ccc} 0\angle 0^\circ & 2.304\angle\text{-}125.63^\circ & 2.304\angle\text{-}215.63^\circ \end{array}\right]^T \text{kA}$$

It is observed that the order of deep region is drastically reduced and thus the obtained Robust TLNE model is very compact in computational aspect. Consequently, the Robust TLNE model for AIES is constructed. The ATP data file of Robust TLNE model is shown in Appendix C.3.2.

To achieve the same level of RMS-error% in aerial mode as Robust TLNE, the FDNE model requires 220th-order rational function. This is much higher than the combined order of the Robust TLNE model.

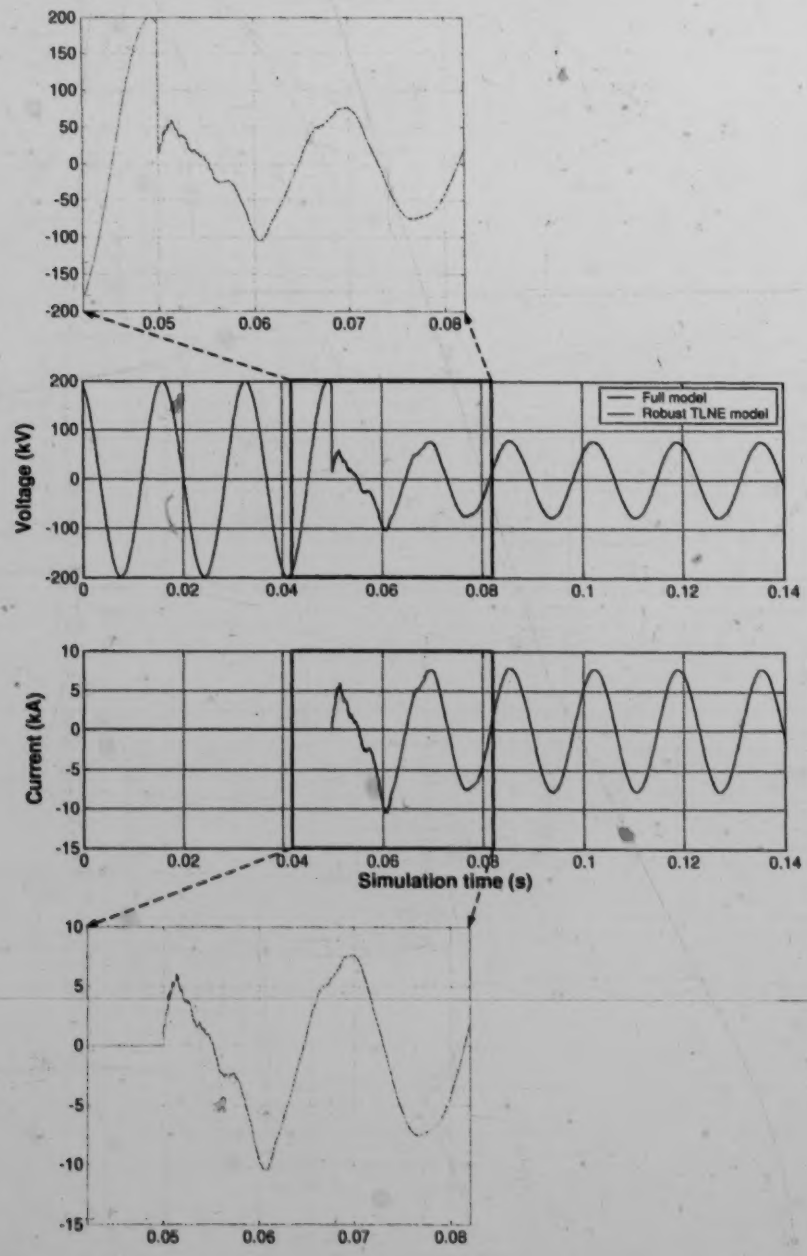


Figure 4.14: AIES Area 50 three-phase to ground fault transient simulation and comparison (phase *A*)

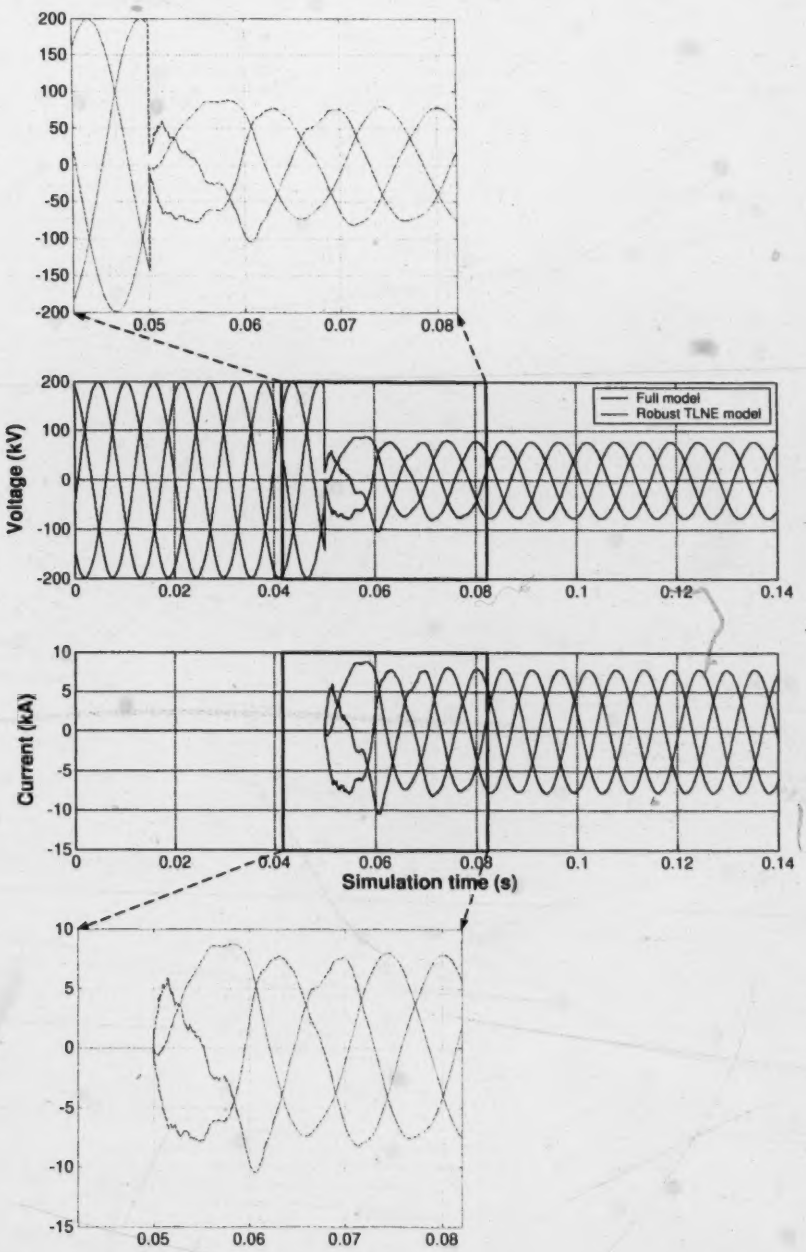


Figure 4.15: AIES Area 50 three-phase to ground fault transient simulation and comparison (three phase)

Transient events	Total time	Full model	Robust TLNE	220th FDNE
Balanced fault	0.15s	17.604s	0.081s	0.352s

Table 4.1: AIES Area 50 computational time comparison at time step size 20μs

4.5 Transient Simulations

Figures 4.14 and 4.15 show phase-*A* and three-phase voltage and current transients at Bus 524 Genesee, respectively. Only phase A is shown, since the same pattern is followed for all three phases. The balanced three-phase to ground fault with 10Ω resistance occurs at 0.05s. Total simulation time is 0.15s. All transients are verified via ATP with time-step size 20μs. Detailed agreement between the full model of the system and the the Robust TLNE model is observed. Computational time is a major saving in Robust TLNE. Table 4.1 shows computational time comparison among full model, Robust TLNE model and FDNE model on the same Pentium IV 1.6GHz computer. The 220th-order FDNE model does demonstrate great savings on computational time. However, computational savings of the Robust TLNE model are much more substantial. As observed from Table 4.1, the robust TLNE model is about 50 times faster than the full model. The more complex the system is, the larger computational saving we obtain in Robust TLNE model. It is also observed that simulation time of the Robust TLNE model is already well below 0.15s. With the precalculation of the inverse of the system conductance matrix in EMTP, more computational savings during real-time simulation of AIES Area 50 can be obtained.

4.6 Summary

In this chapter, the ATP model for a realistic large power system — AIES was developed. The AIES Area 50 Backbone model for electromagnetic transient studies in ATP was constructed based on detailed transmission line information from the TASMo database and equivalent circuits obtained using PSS/E. Then the Robust TLNE model for AIES Area 50 was developed and validated via time-domain simulations of three-phase to ground fault. Great computational saving in Robust TLNE was observed with respect to full model and FDNE model. Based on the computational time, the suitabil-

ity of real-time implementation of large power systems by the Robust TLNE model is verified.

5

Real-Time Simulations Based on the Robust TLNE Model

In the previous chapters, a Robust TLNE model for large power systems has been developed and a realistic large power system — AIES has been modeled by the Robust TLNE. Transient simulations further verified accuracy and computational efficiency of the proposed model. In this chapter, the systems including Example 2 and AIES modeled by TLNE are to be implemented in the Real-Time Simulator in Real-Time eXperimental LABoratory (RTX-LAB), Power Engineering Group, University of Alberta.

5.1 Opal-RT Real-Time Simulator at RTX-LAB

The simulator in RTX-LAB [33] is a cluster-based, fully digital, parallel real-time simulator for power engineering research. It features high-performance computation units with high flexibility and scalability, high-speed communication links, Linux-based Real-Time Operating System (RTOS), fast FPGA (Field-Programmable Gate Array)-based analog and digital I/Os, MATLAB/SIMULINK-based development platform with customizable models, and highly accurate and efficient models and algorithms for power electronic applications.

The hardware and software architecture of the real-time simulator is shown in Fig-

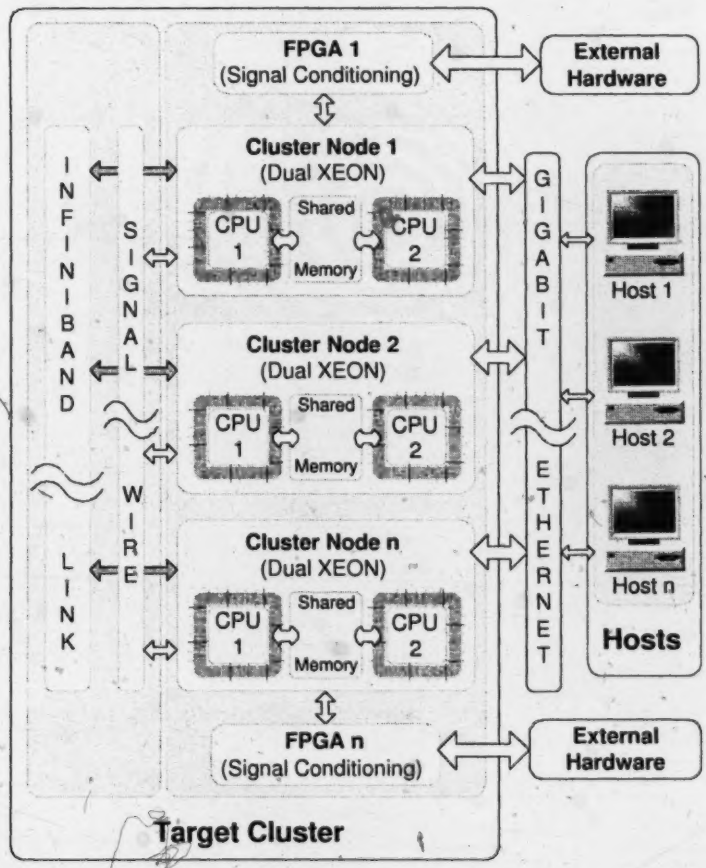


Figure 5.1: Hardware architecture of the RTX-LAB real-time simulator [33]

ures 5.1 and 5.2, respectively. It mainly comprises two groups of computers known as *target nodes* or *targets*, and *hosts*. In addition, high-speed communication links connect targets, as well as hosts and targets. External hardware is connected via FPGA-based analog/digital I/Os. Currently, the simulator has eight targets, each of which is powered by dual 3.0GHz Intel® Xeon™ processors. The two processors or CPUs in one target communicate with each other through shared memory. The targets is also capable of eXtreme High Performance (XHP) mode execution, in which one CPU is dedicated entirely to computation while the other CPU is running RTOS tasks and schedulers. The targets are also required to compile source code generated by MATLAB/SIMULINK Real-Time Workshop (RTW) to executables. The hosts are installed with a real-time

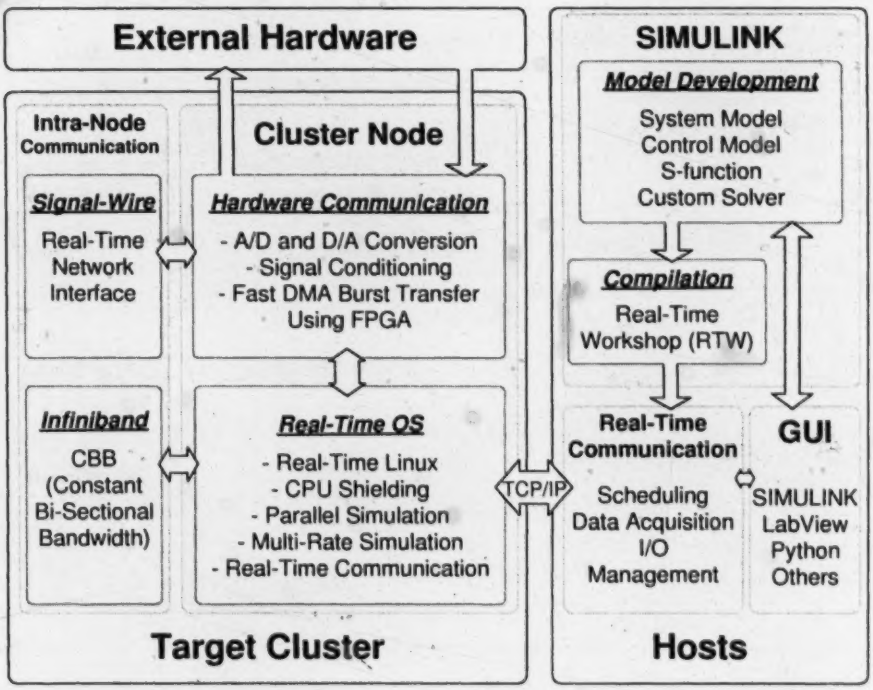


Figure 5.2: Software architecture of the RTX-LAB real-time simulator [33]

interfacing software called RT-LAB provided by Opal-RT Technologies Inc. to coordinate all hardware engaged for the simulation. The hosts are mainly used to create, edit and verify models in SIMULINK, compile SIMULINK blocks into C code by RTW, control and configure real-time simulations in targets, manipulate model parameters in real time as well as acquire real-time simulation results. In case that models are not available in SIMULINK, use-defined S-function block implemented by high-level programming language, *e.g.*, C/C++ and Fortran, *etc.*, can also be incorporated into the models. Again, each host is a high-performance computer which has an 3.0GHz Intel® Pentium® IV CPU to offer fast loading and compilation of the developed models in MATLAB/SIMULINK. Several state-of-the-art computer networking technologies have been utilized to achieve the best communication throughput:

- Shared memory for inter-processor communication in one target. It has the lowest latency.
- InfiniBand architecture for inter-target communication. It has low latency (from

several to several-ten microsecond) depending on communication data size.

- SignalWire which only links adjacent two targets. It has only several-microsecond level of latency.
- Giga-speed Ethernet which mainly connects between targets and hosts, or among hosts.

Based on above introductions, this high-performance real-time simulation platform enables real-time implementation of large power systems modeled by Robust TLNE.

5.2 Implementation of EMTP in MATLAB/SIMULINK

In MATLAB/SIMULINK, due to the fact that Marti's frequency-dependent line model is not available, it is not straightforward to apply Robust TLNE model in SIMULINK. The models in MATLAB/SIMULINK are based on State-Space (SS) approach. However, EMTP uses nodal method. This leads to the solution on implementing EMTP with customized function blocks in SIMULINK. The SIMULINK S-function block, which is implemented by C/C++ and Fortran languages in S-function code format [49], is well-suited in this scenario. For enhanced efficiency, scalability and portability, the S-function is programmed in C++ language. The program mainly includes two parts — reading ATP data files with initializations and simulating obtained electrical networks based on EMTP nodal solutions.

5.2.1 C++ Implementation of EMTP

The C++ is a high-performance Object-Oriented Programming (OOP) language, which features encapsulation, inheritance and polymorphism [52]. Inherited from the nature of OOP, it is efficient, reliable, portable and scalable. The EMTP S-function program fully utilizes the merits of C++ language.

Overview of C++ Classes and Structures for EMTP

In C++, *classes* [52] are the basic elements for OOP programming. The functions inside a class is called *class methods*. An *object* is the instantiation of a class. A class can inherit or be derived from other classes, whose properties will be possessed by the class. The class

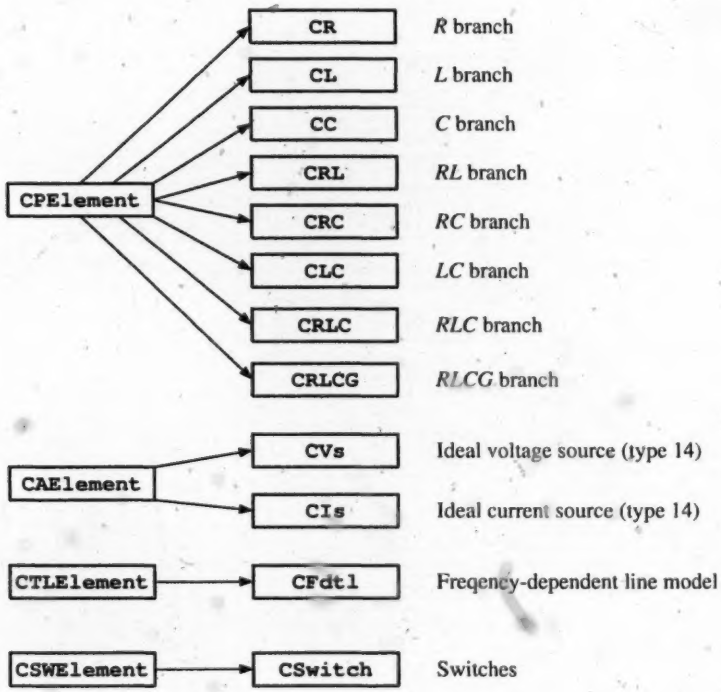


Figure 5.3: C++ class hierarchy for EMTP

for derivations is called *base class*. The C++ concepts can be directly applied in EMTP. Every element of an electrical network is considered as an object instantiated from a class. In a linear network, elements are classified as four base classes — `CPElement`, passive elements such as R, L, C and their combinations; `CAElement`, active elements such as ideal voltage and current sources; `CTLElement`, transmission line models with the consideration of traveling wave effect; `CSWElement`, switches. All realistic element models are derived from the four base classes. Fig. 5.3 shows the class hierarchy for EMTP. Due to the requirement in the Robust TLNE, Marti's frequency-dependent line model is implemented as a class called `CFdtl`, which is derived from `CTLElement`.

One of the most useful features in C++ is polymorphism, in which, a method of a derived class is can be invoked in the base class level by casting the objects of derived class to the ones of base class. For instance, all passive elements derived from `CPElement` class have a method called `update`, which is used to update history current terms in discretized equivalents. In writing the code to update history terms with polymor-

phism, we only need to call the method in base class level instead of calling the same methods in each derived class individually. Thus polymorphism reduces programming complexities and potential bugs. To enable this feature so that the C++ compiler knows exactly the method belonging to which class should be called, a keyword `virtual` is used in the declaration of the methods in base classes.

Class `CPElement` and its Derived Classes

In EMTP, passive elements such as R, L, C mainly have one action to accomplish in each simulation loop — updating history current terms. This is done by calling the method `void update(double&)` in `CPElement` class, where the method parameter is the nodal voltage. In addition, the class must be able to provide the element properties including discretized equivalent conductance, equivalent history current term, and the current that flows through the element. This is realized via methods `double getG()`, `double getIh()` and `double getib()` in `CPElement` class, respectively. The above four class methods form the basic interface of passive elements for the EMTP nodal solution part to invoke. For increased efficiency during simulation, R, L, C, RL, RC, LC, RLC elements are modeled by separate C++ classes derived from `CPElement`. Besides above well-known branch models, $RLCG$ branch model discussed in Section 2.1.4 used in FDNE is also implemented as a separate class called `CRLCG`.

As illustrated in Fig. 2.3, the $RLCG$ branch includes an internal node. Discretization of this branch requires the elimination of the internal node so that obtained equivalent circuit is not only efficient but also can be implemented by derived class of `CPElement`. To realize this approach, both RL and CG blocks are first discretized as an equivalent resistor in series with a history voltage term. The voltage term is only dependent on branch current. Then two equivalent resistors and two voltage terms are merged and grouped. Finally, the circuit is converted to a Norton equivalent — an equivalent resistor in parallel with a history current term, which is suitable for the implementation in a derived class of `CPElement`. Updating current terms requires the calculation of branch current. Therefore, in class method `update`, branch current is first calculated. Since the internal node is eliminated, computational efficiency will increase during matrix computations due to reduced dimension. The same discretization method is also applied to

LC and *RLC* branches for increased efficiency.

During initialization for passive elements in C++, the discretized equivalent resistor and the coefficients of the equation for updating history current terms are calculated and stored based on the class initialization method parameters including the value of all branch elements as well as simulation time-step size. Therefore, those values can be readily accessed during simulation. Appendix D lists the equations for obtaining Norton equivalent and updating history current terms. Those equations are the key to the implementation of passive element classes.

Class `CAElement` and its Derived Classes

The classes for ideal current sources `CIs` and voltage sources `CVs` are derived from `CAElement` class. Three interfaces are provided, which are used to obtain internal conductance by the method `double getG()`, equivalent history current term by the method `double getIeq(double&)`, and branch current by the class method `double getib(double&, double&)`. Current sources are assumed to have very large resistance, *e.g.*, $10^{10}\Omega$. Meanwhile, voltage sources are assumed to have very small resistance, *e.g.*, $10^{-10}\Omega$. The small resistance allows voltage source to convert to Norton equivalent, which is suitable for EMTP. It is verified that the inclusion of the resistance in current and voltage sources has virtually no effect in simulations.

The current simulation time is the parameter of class method `getIeq`, where the instantaneous value of the source is calculated in every time step. Two parameters of `getib` are nodal voltage and current simulation time, respectively. Initialization requires three parameters — frequency, peak magnitude and phase shift.

Class `CTLElement` and its Derived Classes

The `CTLElement` class is used to implement transmission lines with traveling wave effects. Referred to Fig. 2.7, it has six methods — `void update(double&, double&)` for updating history current terms, `double getG()` to obtain discretized equivalent conductance, `double getIhk()` and `double getIhm()` to obtain equivalent current sources of both sides, `getik()` and `getik()` to obtain branch current of both sides. In Robust TLNE model, only frequency-dependent line model is required to be implemented. The class `CFdtl` implements this line model. It strictly conforms

to the equations developed in Section 2.1.5. In addition to the methods defined in the base class, the class method `void updateBkm()` is used to update backward traveling functions $b_k(t)$ and $b_m(t)$ shown in Fig. 2.4 via recursive convolution [24]. The function `void ArrangeF()` is invoked by the class method `update` internally to arrange the forward traveling functions values stored due to traveling delay in every time step. The two parameters of the class method `update` are the nodal voltages of both sides.

Initialization of `CFdtl` class requires poles, residues and constant term of $Z_{eq}(s)$ in (2.22), poles and residues of $P_a(s)$ in (2.32), traveling time τ, and time step size. Then discretized equivalent resistor R_{eq} in (2.27), the coefficients of the equations to update RC parallel blocks in (2.29), and the coefficients of the equations for recursive convolution in (2.36) are calculated and saved.

Class `CSWElement` and its Derived Classes

Class `CSWElement` is the base class for switch classes to derive. In this program, only ideal switch class called `CSwitch` is implemented. The switch class uses very large and very small resistance to simulate switch open and close. Three class methods are provided — `double getG()` for getting the current conductance of the switch, `double getib(double&)` to obtain the branch current flowing through the switch, `void update(bool&)` to update the resistance of the switch according to the current switch state. The parameter of `getib` is the nodal voltage. Logic-type switch state is the parameter of the class method `update`. Logic "1/true" stands for switch open, while logic "0/false" stands for switch close.

C++ Matrix Template Class

The class for matrix manipulation and operation is not available in C++ standard libraries. Thus, a third-party C++ matrix template class, called Matrix TCL Lite by Tech-Soft Pvt. Ltd. [54], is used. The Matrix TCL Lite is only capable of basic linear algebra operations such as matrix addition, substraction, multiplication as well as inversion. Since only matrix multiplication and inversion are required in EMTP nodal solution, the matrix class is well-suited for C++ EMTP implementation.

Type-defined by `matrix<double>`, `CMatrix` is the class used for matrix computations. In C++, taking the advantage of operator overloading [52], the operators for

matrix addition, substraction, multiplication and inversion are "+", "-", "*", and "!", respectively. The C++ header file `matrix.h` is required to be included in the program.

C++ Structures for Storing Electrical Networks

In order to readily and efficiently construct the conductance matrix and current source vector of an electrical network during simulation, several C++ structures are created in this program to store the network in this fashion.

The `CBranch` and `CShunt` structures are the key structures in this EMTP implementation. The `CBranch` structure includes all branch-type elements in the network which connect between two nodes. The `CShunt` structure encompasses all shunt-type elements in the network which connect to ground. All linear elements are able to classified into either type of the structures. The EMTP element objects of the EMTP classes discussed in this section is stored as `void *data` in both `CBranch` and `CShunt`. In addition to the key structures, structures `CIout`, `CVout` and `CSWNode` are used to save current output, voltage output and switch information, correspondingly.

In order to dynamically allocate memory based on the electrical networks to be simulated, `vector` class available in C++ Standard Template Library (STL) [53] is employed. It features high performance, scalability and portability and is seamlessly incorporated with other C++ classes.

Several vectors are defined to provide complete information for the electrical network to be simulated:

- `CNVec`. Type-defined by `vector<string>`, it is used to store all node names in the network.
- `CBVec`. Type-defined by `vector<CBranch>`, it is used to store all branch-type elements defined in structure `CBranch`.
- `CSVec`. Type-defined by `vector<CShunt>`, it is used to store all shunt-type elements defined in structure `CShunt`.
- `CIVec`. Type-defined by `vector<CIout>`, it is used to store all current outputs defined in structure `CIout`.

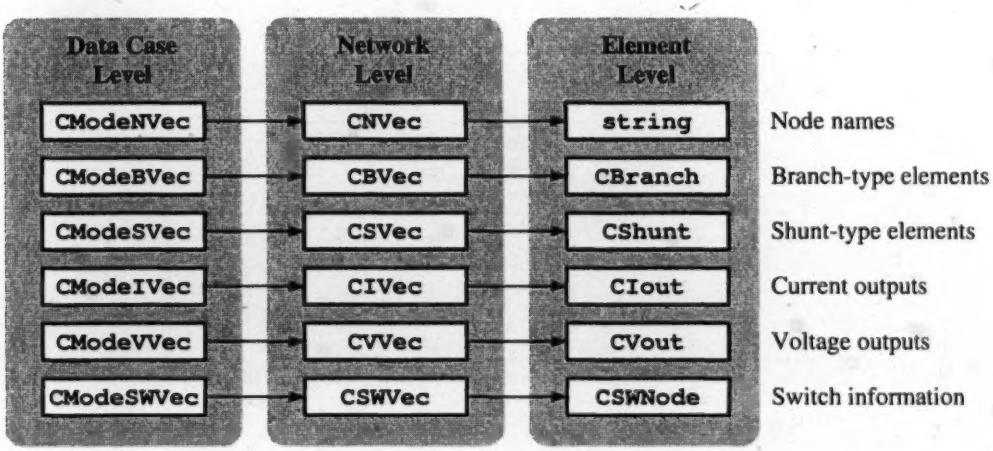


Figure 5.4: C++ data type hierarchy for EMTP

- `CVVec`. Type-defined by `vector<CVout>`, it is used to store all voltage outputs defined in structure `CVout`.
- `CSWVec`. Type-defined by `vector<CSWNode>`, it is used to store all switch information defined in structure `CSWNode`.
- `CMVec`. Type-defined by `vector<CMatrix>`, it is used to store all inverse conductance matrices defined in class `CMatrix`.

All elements and information are properly stored in the above vectors during reading the ATP data files.

To support for multiple data cases, the following vectors are also defined:

- `CModeNVec`, the vector of data type `CNVec`.
- `CModeBVec`, the vector of data type `CBVec`.
- `CModeSVec`, the vector of data type `CSVec`.
- `CModeIVec`, the vector of data type `CIVec`.
- `CModeVVec`, the vector of data type `CVVec`.
- `CModeSWVec`, the vector of data type `CSWVec`.

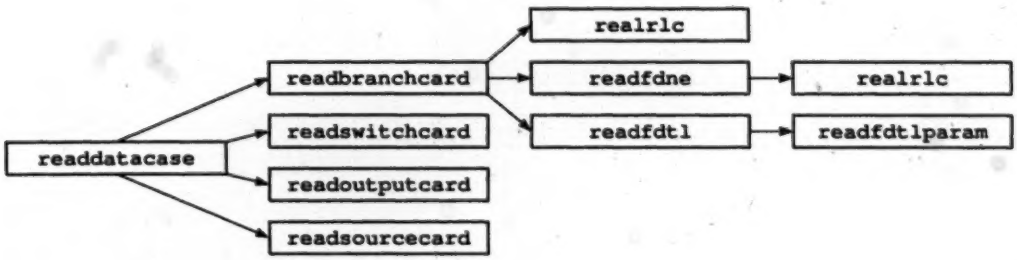


Figure 5.5: C++ function hierarchy for reading ATP data files

- `CModeMVec`, the vector of data type `CMVec`.

Then each data case representing a specific network is saved corresponds to the index in above vectors. The data type hierarchy is shown in Fig. 5.4.

Reading ATP Data Files

In order to generalize the network to be simulated, ATP data files are used as standard input data files for electrical networks. In this fashion, the program is fully adaptable for different electrical networks. Fig. 5.5 illustrates the function hierarchy for reading ATP data files. The main function to read data files is `readdatacase`, inside of which, it calls the corresponding functions to read sorting cards in data files:

- Function `readbranchcard` reads branch cards beginning with `/BRANCH`. Inside this function, three functions are called — `readrlc` to read *RLC* branch, `readfdne` to read FDNE model circuit including *RLCG* branches, and `readfdtl` to read Marti's frequency-dependent line model. The function `readfdne` actually invokes `readrlc` to obtain FDNE model elements. Function `readfdtlparam` reads parameter blocks of Marti's frequency-dependent line model.
- Function `readswitchcard` reads switch cards beginning with `/SWITCH`.
- Function `readoutputcard` reads voltage output cards beginning with `/OUTPUT`.
- Function `readsourcecard` reads source cards beginning with `/SOURCE`.

After invoking the functions for reading ATP data files, all vectors and objects are initialized corresponding to the electrical network to be simulated.

The data file format strictly follows the definitions in ATP Rule Book [44]. Currently, only four types of sorting cards — `/BRANCH`, `/SWITCH`, `/SOURCE` and `/OUTPUT` are supported, inside which, the support for EMTP elements are:

- Linear *RLC* branch with branch current output only.
- Linear *RLCG* branch, which is not originally available in ATP. By using *RLCG* branch, total number of nodes in EMTP can be greatly reduced. Thus computational time is potentially improved. To enable the program to read *RLCG* branch instead of separate *R*, *L*, *C* branches, two guidelines must be followed:
  1. The *RLCG*-branch format is required to follow the format generated by VF for FDNE models. Examples are available in data files in Appendices A.4 and B.4.
  2. In ATP data files, statements `C BEGIN FDNE` and `C END FDNE` are required to immediately precede and immediately follow the FDNE network block, respectively. The first statement is especially important.
- Marti's frequency-dependent line model, whose parameters are generated by ATP LINE CONSTANT routine. Only branch current output is available.
- Type 14 ideal current and voltage sources with current output.
- Ideal switches with current output.

Nodal voltage outputs is supported via `/OUTPUT` card. Current measurement is not explicitly supported. However, this can be done by including an *R* branch of very low resistance and enable branch current output. The sequence of the outputs is that branch current outputs come first and then nodal voltage. The program only support node names of exactly six characters. Node names of less than six characters will lead to unexpected results.

The program is also capable of simulating multiple independent networks simultaneously. This is done by including multiple data cases in one ATP data file with `BEGIN DATA CASE` and `END DATA CASE` statements. This feature is also supported in ATP [44].

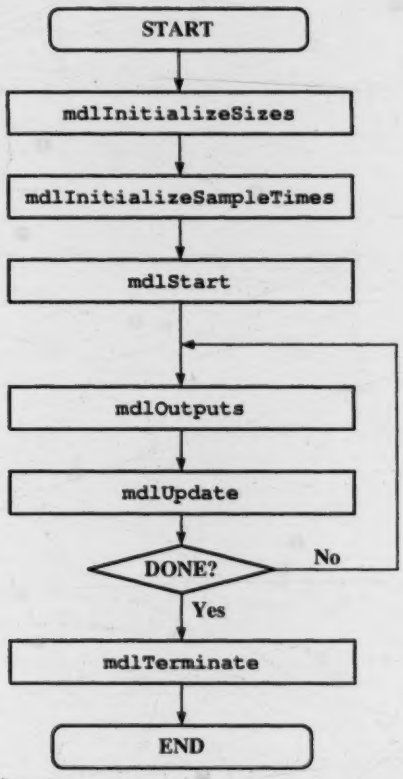


Figure 5.6: SIMULINK S-function flowchart for EMTP

Simulation of Robust TLNE model of three-phase systems is only done through Clarke's transformation of study zone from phase domain to modal domain. However, this is not suitable for general scenarios due to the fact that the study zone may include nonlinear and time-variant elements. Thus, it is a future work to integrate direct three-phase solution method [1] into the EMTP program so that it is suitable for any generic case of transient studies in three-phase systems with the Robust TLNE model.

Limitations on supporting other elements can be gradually reduced by including more types of network elements and incorporating more functionalities in the program in the future, since the C++ code is written in a flexible and scalable way.

5.2.2 SIMULINK S-Function Program Implementing EMTP

The SIMULINK S-Function block is invoked in every simulation loop during simulation. As shown in Fig. 5.6 flowchart, the essential functions in C++ S-function to be used are the follows.

Function `static void mdlInitializeSizes(SimStruct *S)`

This is the function to specify the basic characteristics of the block, such as how many inputs, outputs, the port width of each input and output, and the number of parameters of the S-function block. In this case, one input, one output and two parameters are defined. The input corresponds to switches in the network to be simulated — "1" for switch close, "0" for switch open. This logic is different from the one used in the class method `update` of the `CSwitch` class explained in the previous section. Therefore the logic NOT operating is applied to correct the discrepancies. The output is voltage and current outputs requested in data files for display. Default port width for input and output is both 3, which can be modified by `MAXINPUTPORTWIDTH` and `MAXOUTPUTPORTWIDTH` macros defined in the beginning of the program, if the port width is not sufficient. Two parameters are used in the block — sample time/time-step size and the full file name of the ATP data file for real-time simulation.

Function `static void mdlInitializeSampleTimes(SimStruct *S)`

Sample time of the S-function block is initialized here, which is retrieved from the first S-function parameter by C function `mxGetScalar(ssGetSFcnParam(S, 0))`.

Function `static void mdlStart(SimStruct *S)`

All initializations of EMTP models and matrices are placed in this function. Via functions in Fig. 5.5, ATP data files are read and interpreted, and corresponding models are initialized. Moreover, in order to reduce computational time in the simulation loop, all matrices corresponding to the switching states are built and their inverses are precalculated via `CMatrix` class and saved for future access in simulation loop.

Function `static void mdlOutputs(SimStruct *S, int_T tid)`

This is the function invoked in every simulation loop. Outputs requested in ATP data file are sent to S-function block output. The outputs are calculated in `mdlUpdate` function discussed below.

Function `static void mdlUpdate(SimStruct *S, int_T tid)`

This is the main function implementing EMTP. Invoked in each time step, the function first reads switch states from the S-function block input and retrieves matching inverse system conductance matrix precalculated during initialization in `mdlStart`, then obtains current source vector, calculates nodal voltage vector and updates the history terms (current sources) of EMTP models, and finally calculates outputs requested in the data file and saves them to be used by the function `mdlOutputs` for S-function block output.

Function `static void mdlTerminate(SimStruct *S)`

In this function, memory blocks allocated for storing EMTP models are freed to ensure no memory leakage in the C++ program.

5.2.3 Working with EMTP S-Function Block in Real-Time Simulator

To incorporate EMTP S-function block into RTX-LAB simulator, the following files are required to transfer to targets in RT-LAB: `emtp.cpp`, `emtp.h`, `matrix.h` and the ATP data file for real-time simulation. Together with existing C code generated by RTW, the C++ code is compiled by GCC/G++ in targets to generate real-time executables. However, the S-function program is only able to run in a single target. Implementation of distributed and parallel computation for EMTP is one of the future works to extend in this C++ program.

The complete C++ source code of S-function program is listed in Appendix E. To compile the source code into the MEX-function (executable for MATLAB with extension `.dll` in Microsoft Windows platform), the following command is used in MATLAB:

```
>> mex -g emtp.cpp
```

or

```
>> mex emtp.cpp
```

The `-g` option is used to include debugging information in the MEX-function. More information on debugging SIMULINK S-function is available in [49].

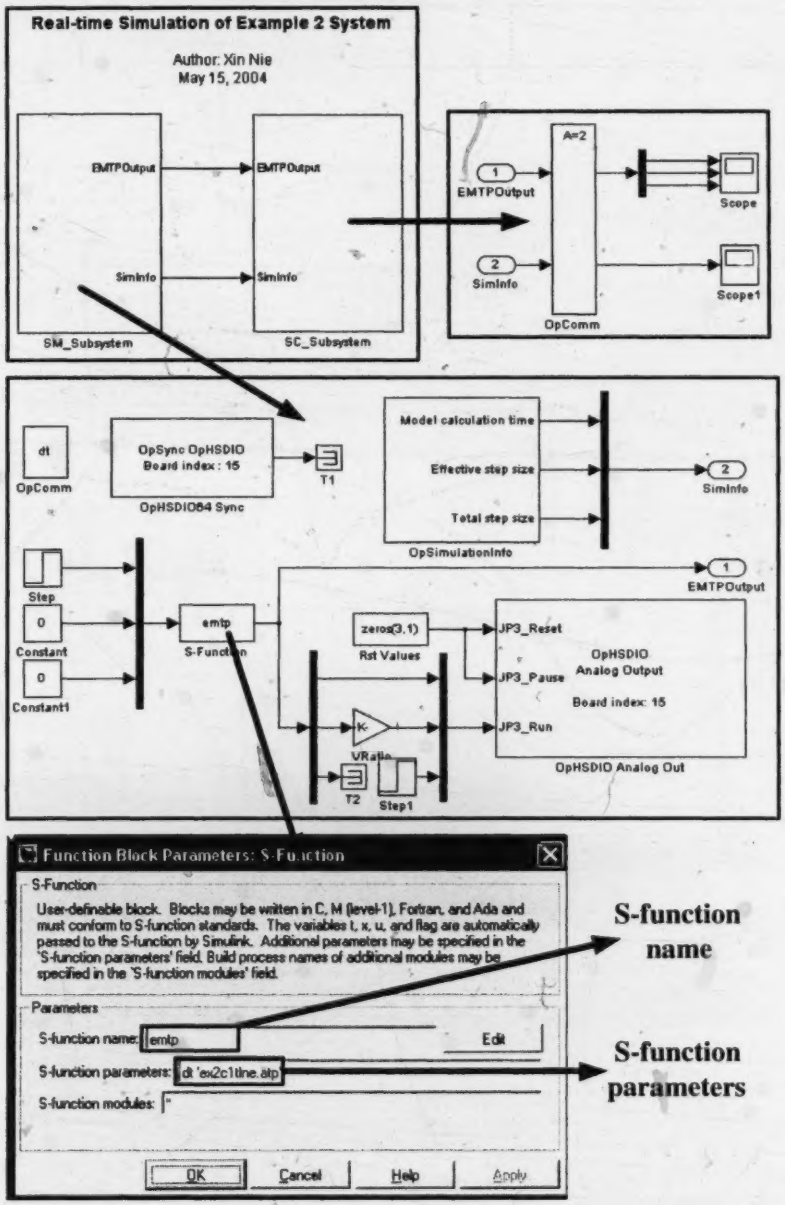


Figure 5.7: Example 2 SIMULINK diagram for S-function based EMTP

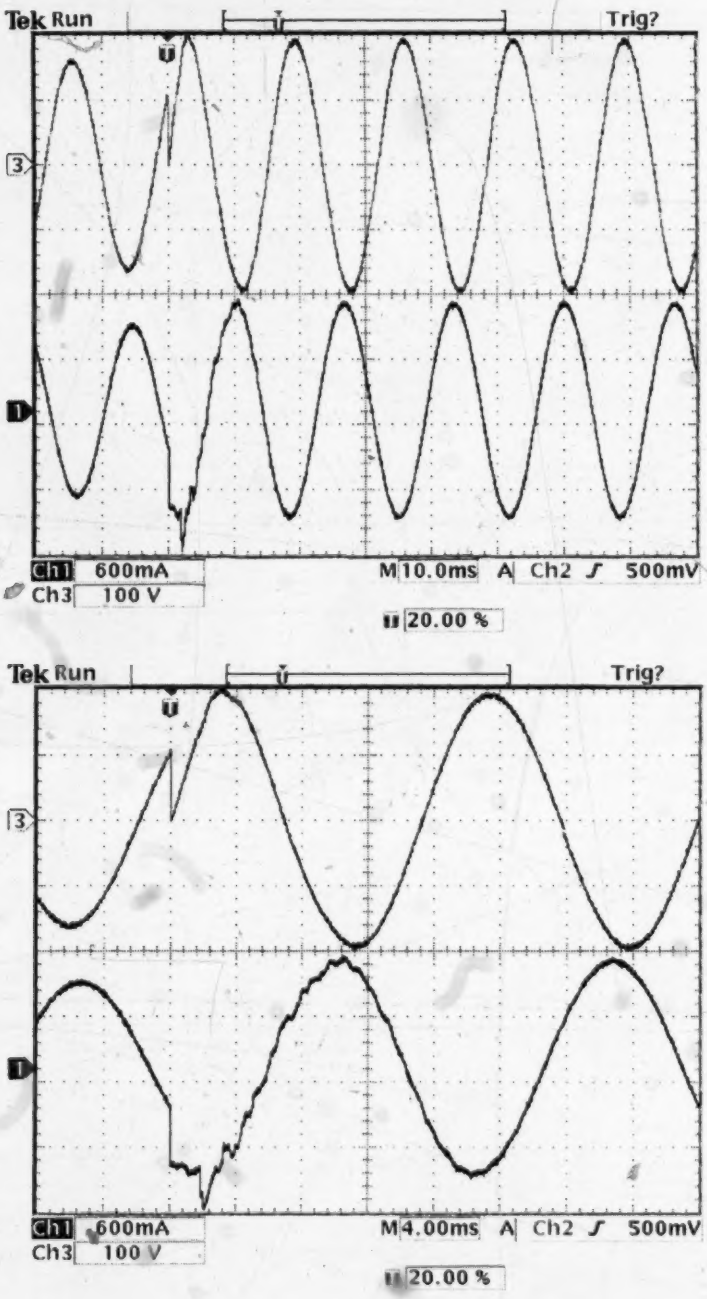


Figure 5.8: Example 2 real-time simulation result (×1k) of capacitor switching (phase *A*)

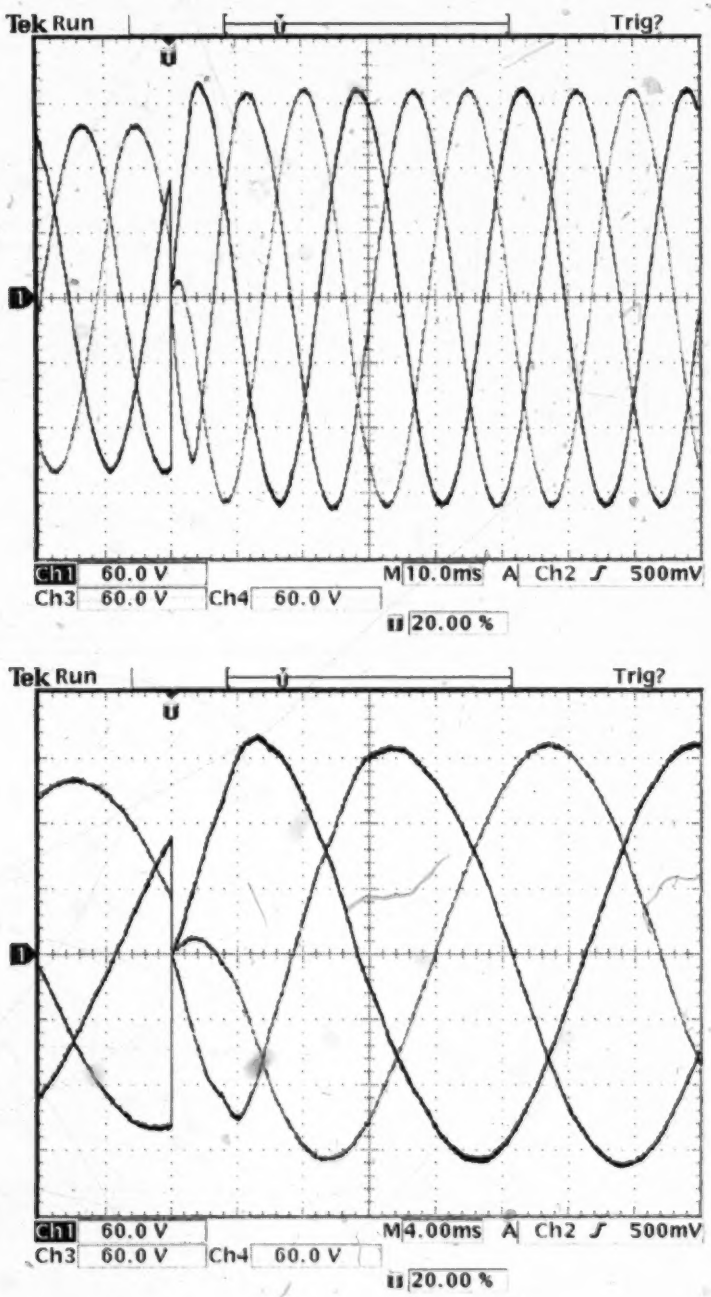


Figure 5.9: Example 2 real-time simulation voltage result (×1k) of capacitor switching (three-phase)

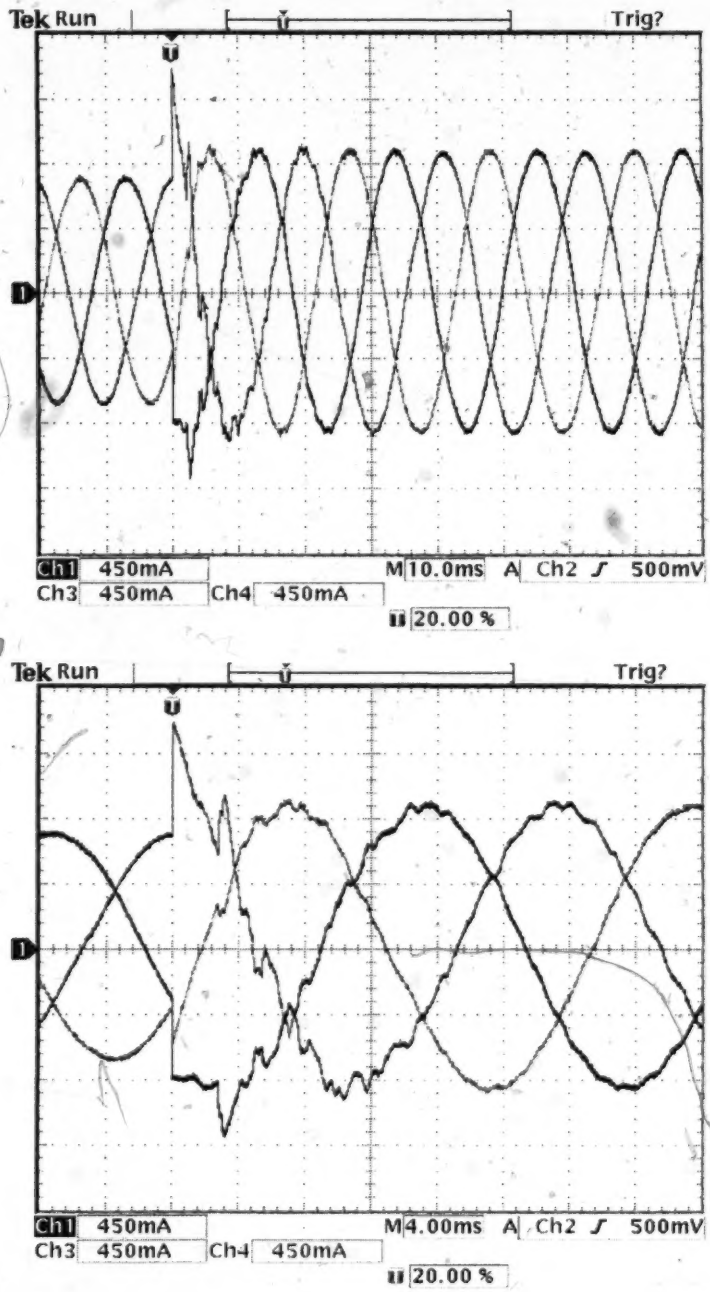


Figure 5.10: Example 2 real-time simulation current result (×1k) of capacitor switching (three-phase)

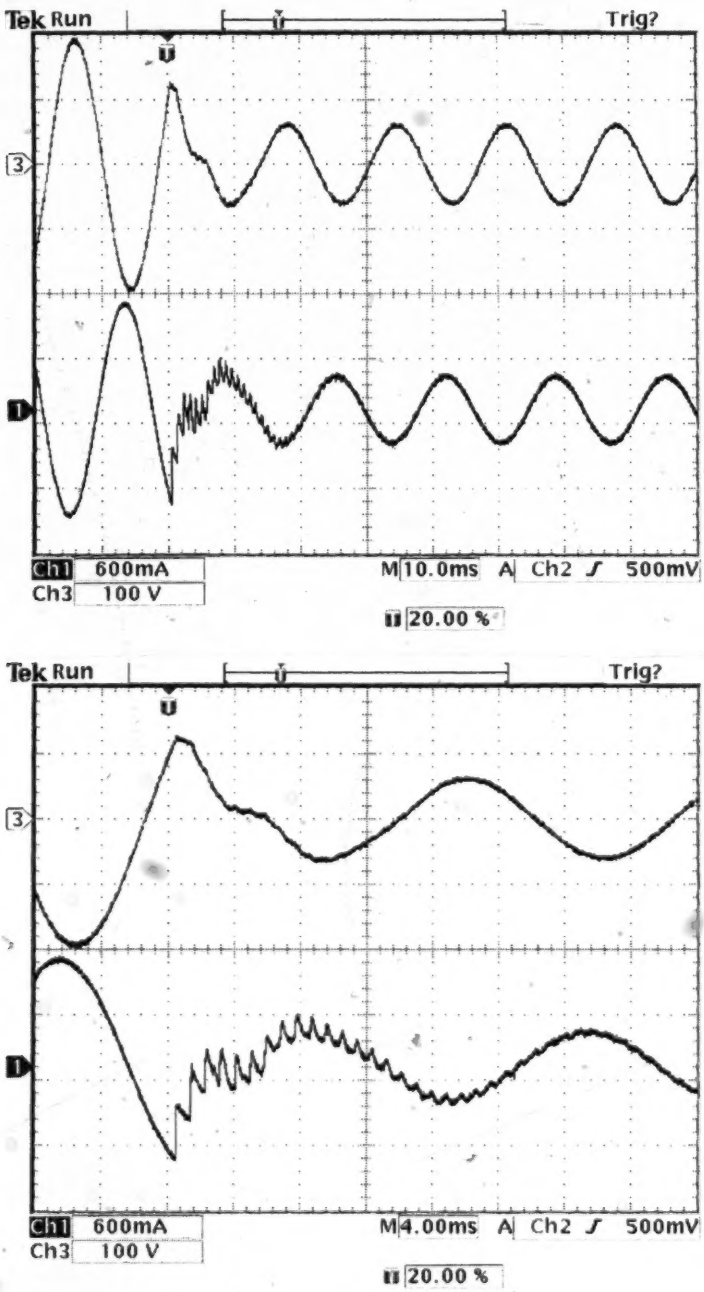


Figure 5.11: Example 2 real-time simulation result (×1k) of balanced fault (phase *A*)

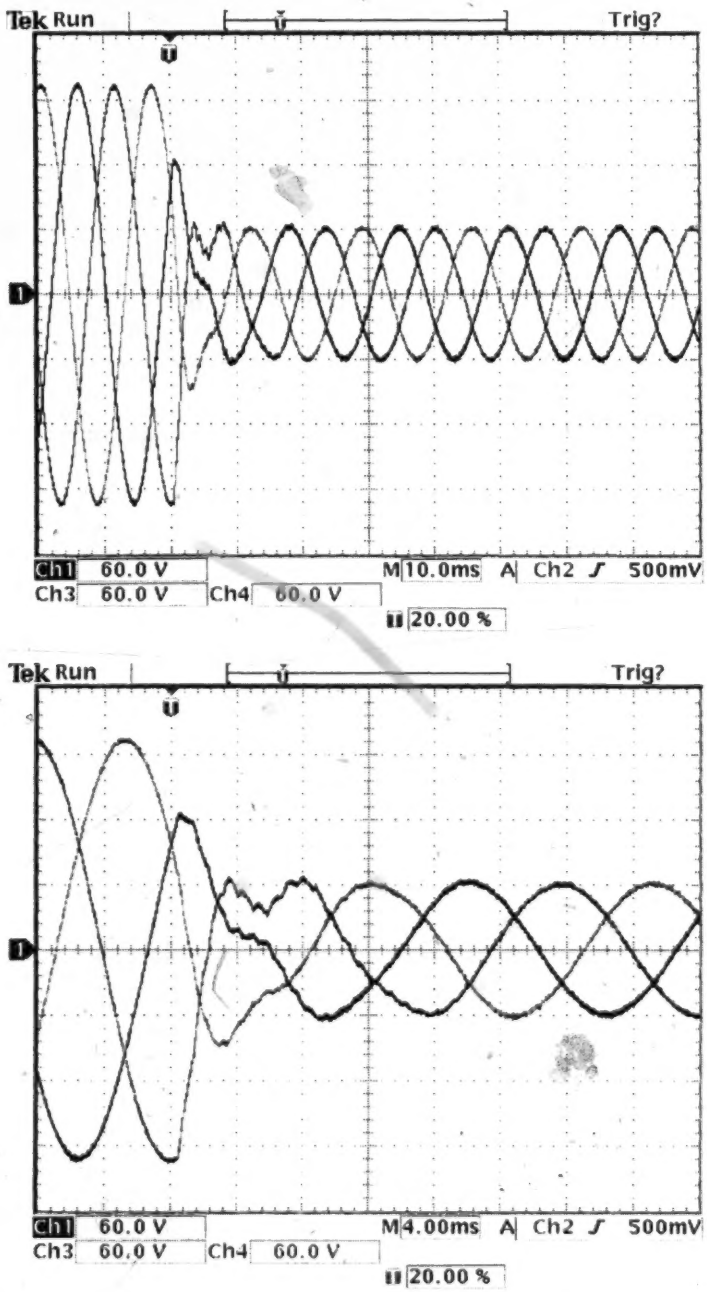


Figure 5.12: Example 2 real-time simulation voltage result (×1k) of balanced fault (three-phase)

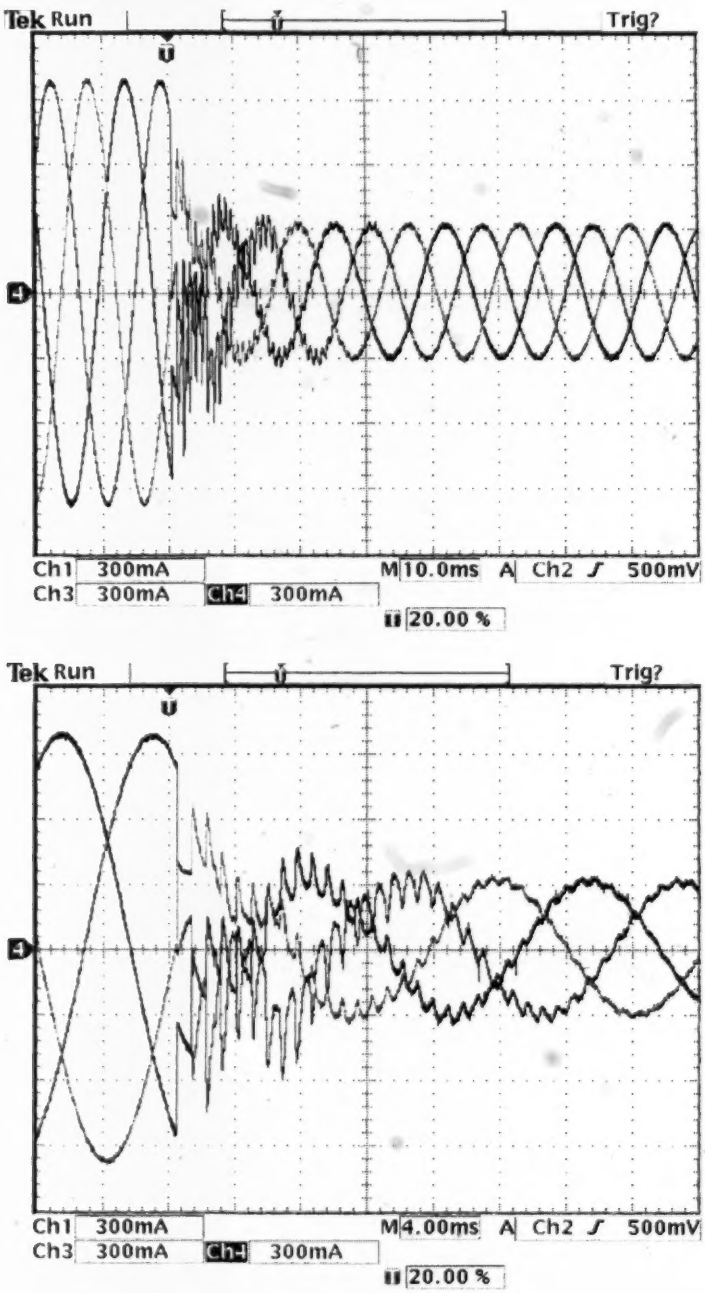


Figure 5.13: Example 2 real-time simulation current result (×1k) of balanced fault (three-phase)

5.3 Real-Time Simulation of Example 2

The system of Example 2 in Section 3.3 is implemented in RTX-LAB real-time simulator at 40μs time step size. Fig. 5.7 illustrates how the diagram is built in SIMULINK for the real-time implementation of the system. Since two events — capacitor switching and balanced three-phase to ground fault use different ATP data files (`ex2c1tlne.atp` and `ex2c2tlne.atp` listed in Appendix B), corresponding file names are entered in the "S-function parameter" field, as shown in Fig. 5.7. Figures 5.8 through 5.10 show capacitor switching (the first transient event) waveforms captured by oscilloscope of phase-A voltage and current, three-phase voltage, and three-phase current, respectively. Figures 5.11 through 5.13 show three-phase ground fault (the second transient event) waveforms captured by oscilloscope of phase-A voltage and current, three-phase voltage, and three-phase current, respectively. The waveforms are indistinguishable to the corresponding off-line simulation results shown in Figures 3.18 through 3.21.

5.4 Real-Time Simulation of AIES

The Robust TLNE model for AIES Area 50 is efficient and suitable for real-time simulation. The real-time simulation diagram in SIMULINK is shown in Fig. 5.14. The file name `aies50tlne.atp` is supplied in the second S-function parameter. Time-step size of 40μs is achieved during real-time simulation. Fig. 5.15 through 5.17 show transient waveforms captured by oscilloscope of phase-A voltage and current, three-phase voltage, and three-phase current, respectively. They are also indistinguishable to the off-line simulation results of AIES Area 50 in Figures 4.14 and 4.15. This case study fully verifies the efficiency and accuracy of the Robust TLNE model and its suitability of real-time simulation of large power systems as well as the efficiency and accuracy of the C++ SIMULINK S-function program implementing EMTP.

5.5 Summary

This chapter introduced the hardware and software architecture of a versatile and fully digital real-time simulator based on the MATLAB/SIMULINK development platform in RTX-LAB, Power Engineering Group, University of Alberta. A C++ S-function im-

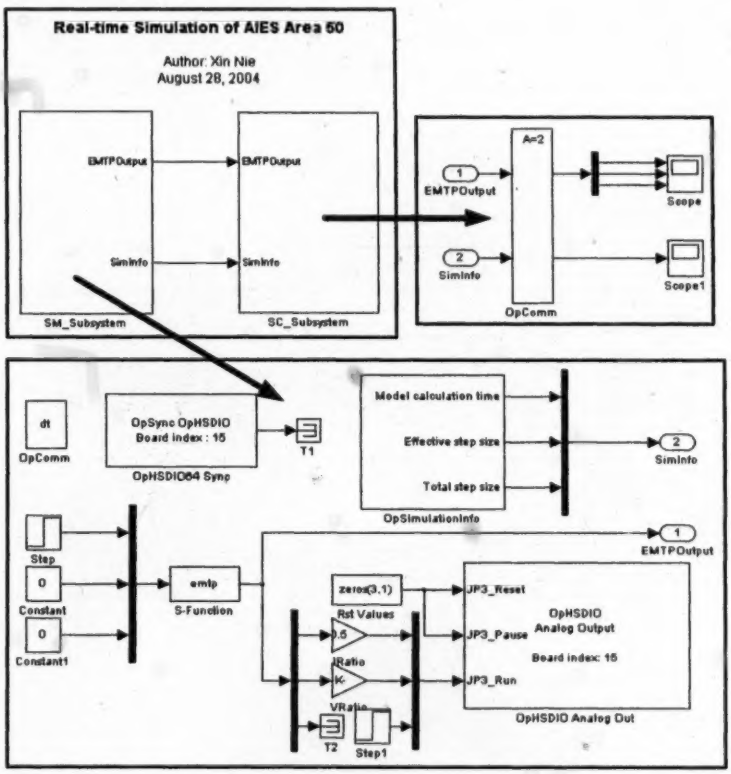


Figure 5.14: AIES Area 50 SIMULINK diagram for S-function based EMTP

plementation of EMTP in SIMULINK, including C++ classes for network elements, C++ structures for EMTP, and C++ functions to read ATP data files, is explained in detail. The merits of the S-function program are the OOP modeling of electrical network elements, the capability of reading ATP data files, the ability to dynamically allocate the memory for different networks, and the efficiency in computation using C++ `vector` class. Then the systems of Example 2 in Chapter 3 and a realistic large power system — AIES in Chapter 4 are implemented in real time, further validating the computational efficiency and accuracy of the Robust TLNE model.

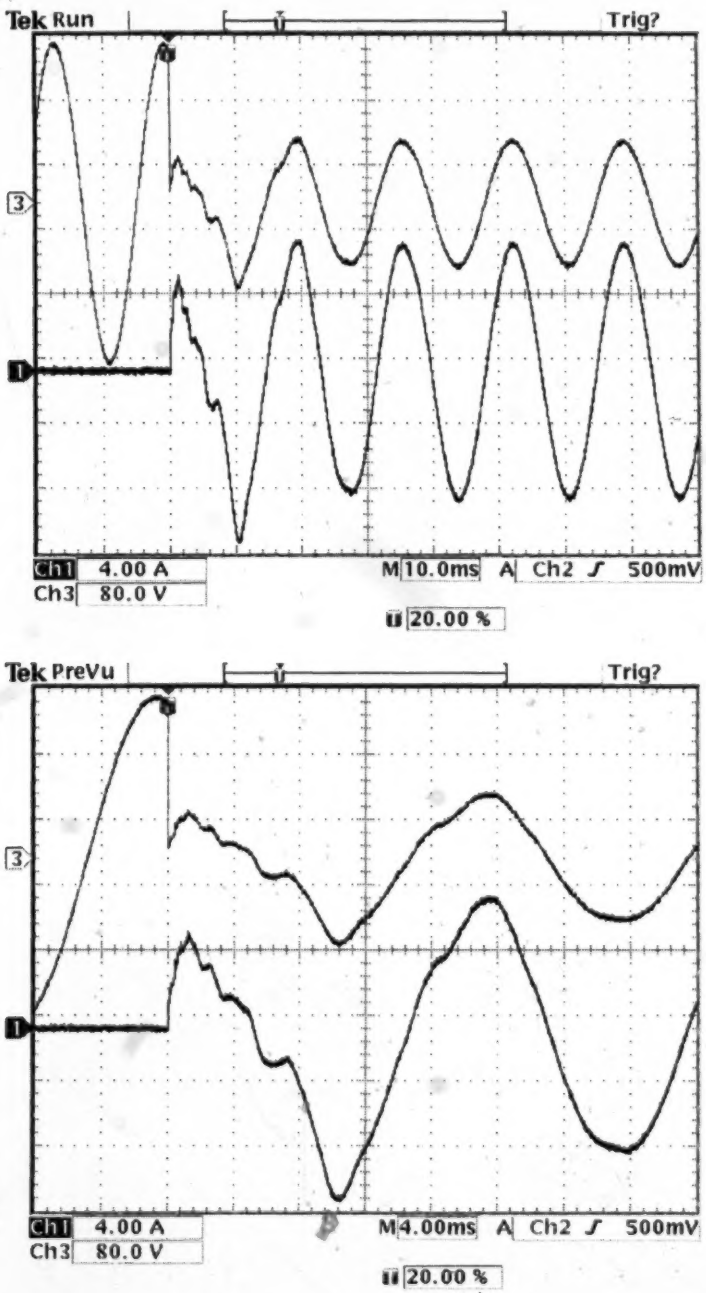


Figure 5.15: AIES Area 50 real-time simulation result ($\times$1k) of balanced fault (phase A)

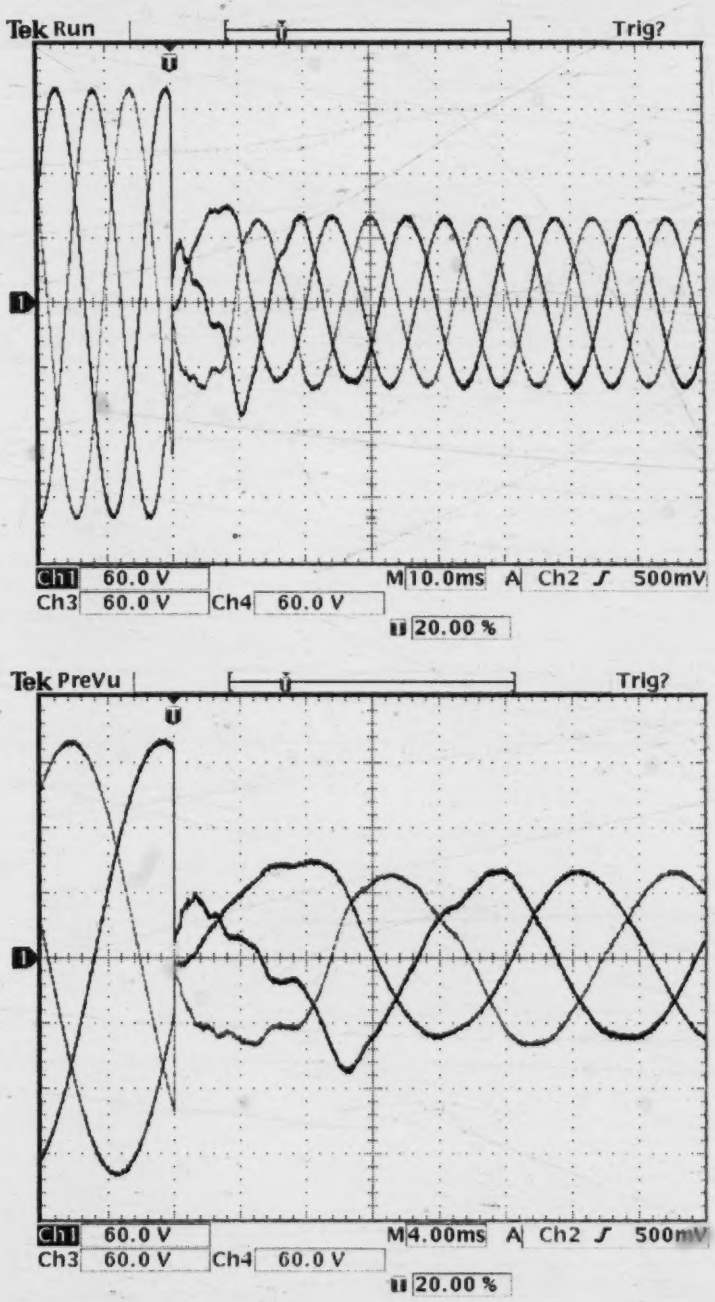


Figure 5.16: AIES Area 50 real-time simulation voltage result (×1k) of balanced fault (three phase)

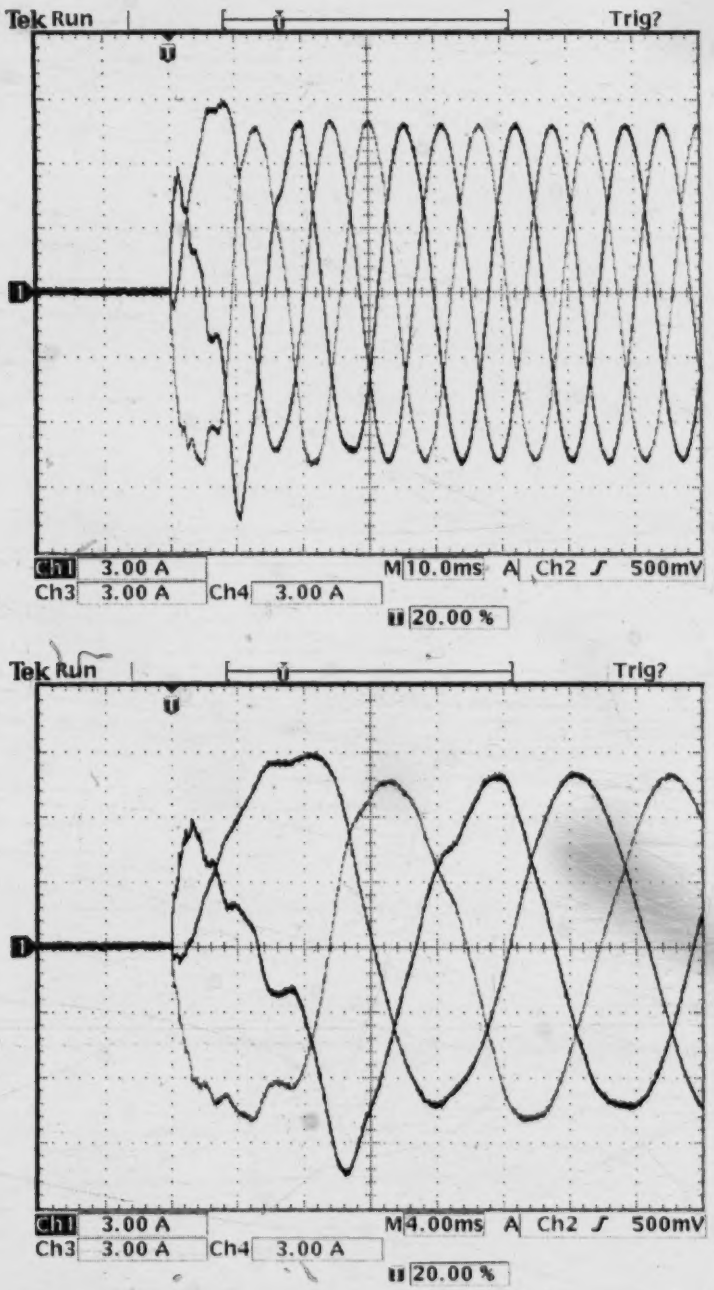


Figure 5.17: AIES Area 50 real-time simulation current result (×1k) of balanced fault (three phase)

6

Conclusions and Future Work

This thesis has proposed a new systematic approach for constructing a TLNE for external systems suitable for electromagnetic transient simulation. With full-order VF, GAs, CLLSQ optimization, and accurate low-order line parameter fitting routine, the generated low-order model is of high accuracy compared to its full model in frequency domain. The merits of this method are its robustness in terms of stability and passivity, its accuracy in not only transient frequencies but also at DC and power frequency, and its optimal deep region order determination feature.

Three detailed systems including a realistic system — the Alberta Interconnected Electric System (AIES) have been model by Robust TLNE and simulated in ATP for verification. Time-domain simulation results with respect to the original system in ATP illustrate the accuracy and computational efficiency of the proposed approach.

Real-time implementation of Robust TLNE models of AIES in RTX-LAB real-time digital simulator is carried out via MATLAB/Simulink C++ S-function. The S-function utilizes the merits of C++ language to ensure high efficiency, scalability and portability of the program. Low time-step size is achieved during real-time simulation with identical results compared to off-line simulation.

The main contributions of this thesis can be summarized as follows:

1. In the modeling of Robust TLNE, this thesis provides that

- Application of real-time Marti's frequency-dependent line model.
- A common set of poles based low-order deep region FDNE model.
- Application of Genetic Algorithms (GAs) with constraints in obtaining first approximations.
- Compensations in GAs for increased accuracies.
- A numerically stable and more efficient CLLSQ optimization algorithm.
- Ensuring accuracies of frequencies at DC of the obtained equivalent.
- Optimal deep region order determination.
- The model with optimal order, for both surface layer and deep region.
- Procedures to construct Robust TLNE model.

2. In constructing an EMTP model for AIES, this thesis contributes

   - An accurate EMTP model for AIES Area 50 Backbone in which all major 240kV transmission lines are represented with frequency dependence.
   - Transferring PSS/E power flow data along with Transmission Alberta System Models (TASMo) database into EMTP model.
   - Procedures to build Robust TLNE for large power systems.

3. In real-time digital simulation, the thesis gives

   - A MATLAB/SIMULINK C++ S-function based EMTP program with the capability of reading ATP data files.
   - Implementation of Marti's frequency dependent line model as well as FDNE including *RLCG* branch in C++.
   - Implementation of *RLCG* branch in EMTP for increased computational efficiency.
   - A collection of C++ class libraries for EMTP.
   - A collection of C++ libraries capable of reading ATP data files.
   - A MATLAB/SIMULINK approach in solving nodal equations of EMTP.

- Real-time digital simulation of large power systems implemented in RTX-LAB real-time simulator.
- A practical example to analyze electromagnetic transients in AIES.

The following topics are proposed for future work:

- Built a detailed EMTP model for AIES and implement the model in the RTX-LAB real-time simulator. Existing utilities such as SEQ2ATP for ATP and E-Tran for PSCAD/EMTDC are capable of converting data of IEEE standard power-flow format to corresponding EMTP packages. This facilitates the implementation of the whole AIES in EMTP.

- Include considerations on un-transposed transmission lines in surface layer. The study zone requires accurate representation of electrical elements in power systems. If more accurate line models are taken into account in surface layer, the surface layer is a part of the study zone and the study zone can be smaller.

- Enforce passivity on passive Robust TLNE model in the entire frequency range. In the proposed Robust TLNE, passivity criterion is only validated in limited frequency range and discrete frequency points during model generation, which is still potential stability issue for TLNE model.

- Initiate more comprehensive transient studies on AIES based on Robust TLNE model. Only a phase to ground fault is studies in this thesis. More transient studies can be done with the existing Robust TLNE model for AIES.

- Integrate direct support for three-phase systems in the EMTP S-function program. Transient studies are only limited to single-phase systems in the program. Three-phase systems are only supported via conversion to single-phase systems. Therefore, it is necessary to implement the support for generic transient studies of three-phase systems.

- Implement the Marti's frequency-dependent line model in MATLAB/SIMULINK. The Robust TLNE model can be directly incorporated into SIMULINK with State-Space solutions if the frequency-dependent line model is available in SIMULINK.

- Incorporate more network elements and functionalities into the C++ EMTP S-function program to support models such as nonlinear devices, power electronics, induction machines and control systems.

- Realize distributed and parallel computation for the EMTP in S-function program. Successful achievements has been noticed in partitioning power system networks for parallel simulation in EMTP [34]. This is done via transmission line models.

Bibliography

[1] H. W. Dommel, "Digital computer solution of electromagnetic transients in single and multiphase networks", *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-88, No. 4, April 1969, pp. 388-399.

[2] H. W. Dommel, "Nonlinear and time-varying elements in digital simulation of electromagnetic transients", *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-90, No. 6, November/December 1971, pp. 2561-2567.

[3] J. R. Marti, J. Lin, "Suppression of numerical oscillations in the EMTP", *IEEE Trans. on Power Systems*, Vol. 4, No. 2, May 1989, pp. 739-747.

[4] CIGRÉ WG13-05 III, "Transmission line representation of energization and re-energization studies for complex feeding networks", *Electra*, Vol. 62, January 1979, pp. 45-78.

[5] A. Semlyen, M. R. Iravani, "Frequency domain modeling of external systems in an electromagnetic transients program", *IEEE Trans. on Power Systems*, Vol. 8, No. 2, May 1993, pp. 527-533.

[6] M. Kizilcay, "Low-order network equivalents for electromagnetic transients studies", *European Trans. on Electrical Power Engineering*, Vol. 3, No. 2, March/April 1993, pp. 123-129.

[7] M. Kizilcay, "Computation of switching transients using low-order multi-port network equivalents", *Proc. of IPST'97*, Seattle, WA., June 1997, pp. 125-130.

[8] W. C. Boaventura, A. Semlyen, M. R. Iravani, A. Lopes, "Robust sparse network equivalent for large systems: part I-methodology", *IEEE Trans. on Power Systems*, Vol. 19, No. 1, Feb. 2004, pp. 157 - 163.

[9] W. C. Boaventura, A. Semlyen, M. R. Iravani, A. Lopes, "Robust sparse network equivalent for large systems: part II-performance evaluation", *IEEE Trans. on Power Systems*, Vol. 19, No. 1, Feb. 2004, pp. 157 - 163.

[10] A. S. Morched, V. Brandwajn, "Transmission network equivalents for electromagnetic transients studies", *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-102, No. 9, September 1983, pp. 2984-2994.

[11] A. S. Morched, J. H. Ottevangers, L. Marti, "Multi-port frequency dependent network equivalents for the EMTP", *IEEE Trans. on Power Delivery*, Vol 8, No. 3, July 1993, pp. 1402-1412.

[12] V. Q. Do, M. M. Gavrilovic, "An iterative pole removal method for synthesis of power systems equivalent networks", *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-103, No. 8, August 1984, pp. 2065-2070.

[13] V. Q. Do, M. M. Gavrilovic, "A synthesis method for one-port and multiport equivalent networks for analysis of power systems transient", *IEEE Trans. on Power Systems*, Vol. PWRD-1, No. 2, April 1986, pp. 103-113.

[14] N. R. Watson, J. Arrillaga, "Frequency-dependent AC system equivalents for harmonic and transient converter simulation", *IEEE Trans. on Power Delivery*, Vol. 3, No. 3, July 1988, pp. 1196-1202.

[15] A. Abur, H. Singh, "Time domain modeling of external systems for electromagnetic transients programs", *IEEE Trans. on Power Systems*, Vol. 8, No. 2, May 1993, pp. 671-677.

[16] H. Singh, A. Abur, "Multi-port equivalencing of external systems for simulation of switching transients", *IEEE Trans. on Power Delivery*, Vol. 10, No. 1, January 1995, pp. 374-380.

[17] B. Gustavsen, A. Semlyen, "Rational approximation of frequency domain responses by vector fitting", *IEEE Trans. on Power Delivery*, Vol. 14, No. 3, July 1999, pp. 1052-1061.

[18] B. Gustavsen, A. Semlyen, "Enforcing passivity for admittance matrices approximated by rational functions", *IEEE Trans. on Power Systems*, Vol. 16, No. 1, February 2001, pp. 97-104.

[19] B. Gustavsen, "Computer code for rational approximation of frequency dependent admittance matrices", *IEEE Trans. on Power Delivery*, Vol. 17, No. 4, October 2002, pp. 1093-1098.

[20] B. Gustavsen, A. Semlyen, "A robust approach for system identification in the frequency domain", *IEEE Trans. on Power Delivery*, Accepted for future publication.

[21] M. Abdel-Rahman, A. Semlyen, M. R. Iravani, "Two-layer network equivalent for electromagnetic transients", *IEEE Trans. on Power Delivery*, Vol. 18, No. 4, October 2003, pp. 1328-1335.

[22] M. Abdel-Rahman, *Frequency Dependent Hybrid Equivalents of Large Networks*, Ph.D. Thesis, University of Toronto, 2001.

[23] X. Nie, V. Dinavahi, "A robust two-layer network equivalent for transient studies", *Proc. of IPST'05*, Montréal, Canada, June 19-23 2005.

[24] A. Semlyen, A. Dabuleanu, "Fast and accurate switching transient calculations on transmission lines with ground return using recursive convolutions", *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-94, No. 2, March/April 1975, pp. 561-571.

[25] J. R. Marti, "Accurate modeling of fequency-dependent transmission lines in electromagnetic transient simulations", *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-101, No. 1, January 1982, pp. 147-155.

[26] F. Castellanos, J. R. Marti, "Full frequency-dependent phase-domain transmission line model", *IEEE Trans. on Power Systems*, Vol. 12, No. 3, August 1997, pp. 1331-1339.

[27] T. Noda, N. Nagaoka, A. Ametani, "Phase domain modeling of frequency-dependent transmission lines by means of an ARMA model", *IEEE Trans. on Power Delivery*, Vol. 11, No. 1, January 1996, pp. 401-411.

[28] T. Noda, N. Nagaoka, A. Ametani, "Further improvements to a phase domain ARMA line model in terms of convolution, steady-state initialization and stability", *IEEE Trans. on Power Delivery*, Vol. 12, No. 3, July 1997, pp. 1327-1332.

[29] A. B. Fernandes, W. L. A. Neves, "Phase-domain transmission line models considering frequency-dependent transformation matrices", *IEEE Trans. on Power Delivery*, Vol. 19, No. 2, April 2004, pp. 708-714.

[30] A. Semlyen, M. H. Abdel-Rahman, "A state variable approach for the calculation of switching transients on a power transmission line", *IEEE Trans. on Circuits and Systems*, Vol. CAS-29, No. 9, September 1982, pp. 624-633.

[31] B. Gustavsen, A. Semlyen, "Simulation of transmission line transients using vector fitting and model decomposition", *IEEE Trans. on Power Delivery*, Vol. 13, No. 2, April 1998, pp. 605-614.

[32] B. Gustavsen, A. Semlyen, "Combined phase and model domain calculation of transmission line transients based on vector fitting", *IEEE Trans. on Power Delivery*, Vol. 13, No. 2, April 1998, pp. 596-604.

[33] L. Pak, M. O. Faruque, X. Nie, V. Dinavahi, "A versatile cluster-based real-time digital simulator for power engineering research", *Submitted to IEEE Trans. on Power Systems*. July 2005.

[34] J. R. Marti, L. R. Linares, "Real-time EMTP-based transients simulation", *IEEE Trans. on Power Systems*, Vol. 9, No. 3, August 1994, pp. 1309-1317.

[35] A. Semlyen, F. de León, "Computation of electromagnetic transients using dual or multiple time steps", *IEEE Trans. on Power Systems*, Vol. 8, No. 3, August 1993, pp. 1274-1281.

[36] R. M. Mathur, X. Wang, "Real-time digital simulator of the electromagnetic transients of power transmission lines", *IEEE Trans. on Power Delivery*, Vol. 4, No. 2, April 1989, pp. 1275-1280.

[37] L. Marti, "Low-order approximation of transmission line parameters for frequency-dependent models", *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-102, No. 11, November 1983, pp. 3582-3589.

[38] X. Wang, R. M. Mathur, "Real-time digital simulator of the electromagnetic transients of transmission lines with frequency dependence", *IEEE Trans. on Power Delivery*, Vol. 4, No. 4, October 1989, pp. 2249-2255.

[39] N. Zhang, X. Wang, J. F. Eggleston, R. M. Mathur, "Improvements in the realization of a real-time digital simualtor of a power transmission line", *IEE International Conference on AC and DC Power Transmission*, London, U. K., September 17-20 1991, pp. 356-361.

[40] C. Dufour, H. Le-Huy, J. Soumagne, A. E. Hakimi, "Real-time simulation of power transmission lines using Marti model with optimal fitting on dual-DSP card", *IEEE Trans. on Power Delivery*, Vol. 11, No. 1, January 1996, pp. 412-419.

[41] X. Wang, D. A. Woodford, R. Kuffel, R. Wierckx, "A real-time transmission lines model for a digital TNA", *IEEE Trans. on Power Delivery*, Vol. 11, No. 2, April 1996, pp. 1092-1097.

[42] A. B. Fernandes, W. L. A. Neves, "Transmission lines: fitting technique optimization", *Proc. of IPST'97*, Seattle, U. S. A., June 22-26 1997.

[43] A. B. Fernandes, W. L. A. Neves, "Frequency-dependent low order approximation of transmission line parameters", *Proc. of IPST'99*, Budapest, Hungary, June 20-24 1999.

[44] Canadian/American EMTP user group, *Alternative Transient Program Rule Book*, 1999.

[45] W. Scott-Meyer, *EMTP Theory Book*, Bonneville Power Administration, 1984.

[46] M. Kizilcay, "A new branch in the ATP-EMTP high-order, linear admittance model", *ATP News*, 1993.

[47] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, January 1989.

[48] A. Chipperfield, P. Fleming, H. Pohlheim, C. Fonseca, *Genetic Algorithm Toolbox for MATLAB*, Department of Automatic Control and Systems Engineering, Univerisity of Sheffield.

[49] *MATLAB User Guides* The MathWorks Inc., Natick, MA..

[50] P. Kundur, *Power System Stability and Control*, McGraw-Hill, 1994.

[51] J. Duncan Glover, Mulukutla S. Sarma., *Power System Analysis and Design (3rd Edition)* Wadsworth/Thomson Learning, 2002.

[52] B. Eckel, *Thinking in C++, Volume 1: Introduction to Standard C++ (2nd Edition)*, Prentice Hall, 2000.

[53] B. Eckel, C. Allison, *Thinking in C++, Volume 2: Practical Programming (2nd Edition)*, Prentice Hall, 2003.

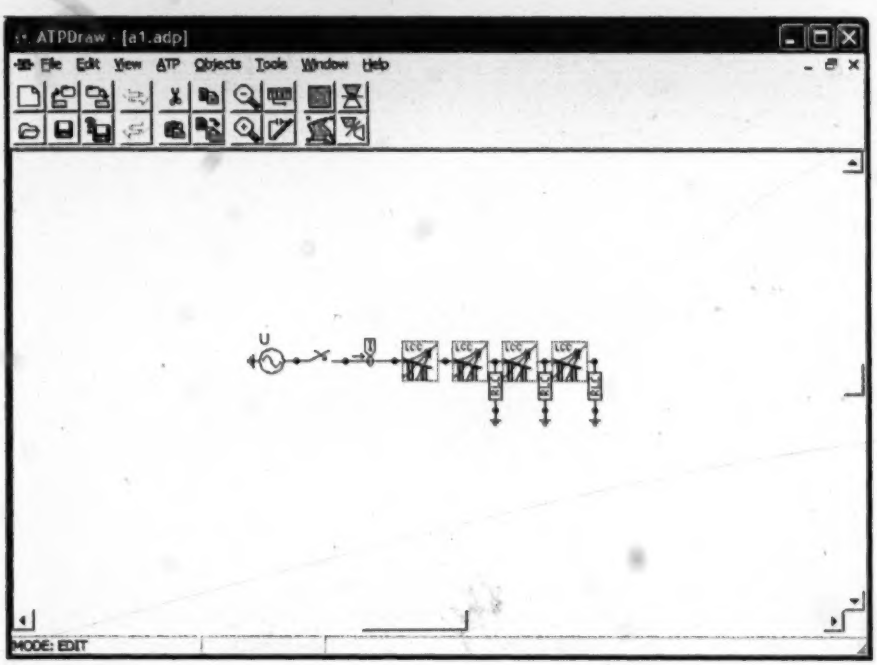
[54] *Matrix TCL Reference Manual*, Techsoft Pvt. Ltd..

A
Example 1

A.1 System Parameters

Elements	Parameters
Source	60Hz, $10\angle 0°$kV
TL1	280km transmission line
TL2	400km transmission line
TL3	220km transmission line
TL4	80km transmission line
Conductor	Two-bundle Drake conductor with 50cm distance
Tower	Height: 15m, ground resistivity: 100Ωm, homogeneous earth assumed
Load1	RL Load: 100Ω, 150mH
Load2	RL Load: 60Ω, 80mH
Load3	RL Load: 120Ω, 100mH

A.2 System Diagram in ATPDraw



A.3 ATP Data File for Full Model

```
BEGIN NEW DATA CASE
C ------------------------------------------------------------
C Xin Nie
C Power Engineering Group
C Dept. of Electrical and Computer Engineering
C University of Alberta
C June 22, 2004
C ------------------------------------------------------------
POWER FREQUENCY                    60.
$DUMMY, XYZ000
C  dT  >< Tmax >< Xopt >< Copt >
   2.E-5      .2
     500      1      0      0      0      0      0      1      0
C        1       2       3       4       5       6       7       8
C 345678901234567890123456789012345678901234567890123456789012345678901234567890
/BRANCH
C < n 1>< n 2><ref1><ref2>< R  >< L  >< C  >
C < n 1>< n 2><ref1><ref2>< R  >< A  >< B  ><Leng><><>0
C Load 1
        XX0006               60.   80.                                        0
C Load 2
        XX0004              100.  150.                                        0
C Load 3
        XX0012              120.  100.                                        0
C TL1
-1A____1XX0002                2.  0.00              -2 1
      18       3.43904357882401600000E+02
   2.75748917329525200E+02   2.09556456539013700E+03  -3.02120107577198200E+02
   1.32522038307429600E+02   5.34206778326990500E+02   7.77786264529477200E+02
   1.49408699070299500E+03   7.42950753414815600E+03   3.11676509575282100E+04
   1.27236036995146400E+05   5.08023732184690700E+05   2.32321224243896200E+06
   5.09434464995741600E+06   3.65432987658662500E+06   2.66960449636756200E+06
   2.78771199529035400E+06   2.85557150987219100E+06   3.28340044978634400E+06
   1.55157367465512800E+00   3.13344553898543100E+00   3.37321729335057800E+00
   4.11467825483795900E+00   8.24190385834596400E+00   1.44386945071827500E+01
   4.50761403850135000E+01   3.97846218782673400E+02   1.72512452765138800E+03
   7.33827995192348500E+03   3.07002204634114500E+04   1.46775367026548200E+05
   6.61870409143599800E+05   1.37066806499039800E+06   1.35751400884485100E+06
   3.13588034329753700E+06   2.94525940129055000E+06   3.70207919969885800E+06
      16       9.92561082855090700000E-04
   1.21611384045855800E-02   3.17820537468208000E-02   1.63475935146936700E-01
   4.50277411304402400E-01   6.24587870467199700E-01   9.62547231237992800E-01
   2.11492778892177200E+00   1.76164573866439500E+01   9.33425314822263600E+01
   5.94613595758890300E+02   1.64245992775248200E+03   2.02960381955791800E+03
   1.56675491715680900E+04   3.56471480032259900E+04   1.59460983021337400E+07
  -1.60017949953998000E+07
   4.47470371651519400E+00   1.17382684613682300E+01   5.99242290097617800E+01
   1.59436483822033800E+02   2.23499770666861100E+02   3.22868571196113200E+02
   3.70637016742645200E+02   7.30820641779627400E+02   1.71955506888262700E+03
   3.96288728577050700E+03   6.25989363073368600E+03   9.88370958996017500E+03
   1.95251223260011500E+04   4.43329805258528100E+04   2.98053414854056900E+04
   2.98351468268911200E+04
  1.00000000
  0.00000000
C TL2
-1XX0002XX0004                2.  0.00              -2 1
      18       3.43904357882401600000E+02
   2.75748917329525200E+02   2.09556456539013700E+03  -3.02120107577198200E+02
   1.32522038307429600E+02   5.34206778326990500E+02   7.77786264529477200E+02
   1.49408699070299500E+03   7.42950753414815600E+03   3.11676509575282100E+04
   1.27236036995146400E+05   5.08023732184690700E+05   2.32321224243896200E+06
   5.09434464995741600E+06   3.65432987658662500E+06   2.66960449636756200E+06
   2.78771199529035400E+06   2.85557150987219100E+06   3.28340044978634400E+06
   1.55157367465512800E+00   3.13344553898543100E+00   3.37321729335057800E+00
   4.11467825483795900E+00   8.24190385834596400E+00   1.44386945071827500E+01
   4.50761403850135000E+01   3.97846218782673400E+02   1.72512452765138800E+03
   7.33827995192348500E+03   3.07002204634114500E+04   1.46775367026548200E+05
   6.61870409143599800E+05   1.37066806499039800E+06   1.35751400884485100E+06
   3.13588034329753700E+06   2.94525940129055000E+06   3.70207919969885800E+06
      12       1.43447271640600000000E-03
   2.24741581214468300E-02   6.14193666189440300E-01   1.73631140036276900E-01
  -1.88996292345958600E+00   1.05136587209149200E+01   1.79732027306505200E+02
   2.80020207640768800E+02   6.47272395784606100E+02   5.42303119579675400E+03
   4.13756337182865700E+05   4.26608490676978000E+07  -4.30811448947022200E+07
   4.40522127150029800E+00   1.16980102042384100E+02   3.42279960201415600E+01
   3.24436046529355400E+02   3.21222961401006300E+02   2.02563796805317100E+03
   2.44012031826036300E+03   3.54015992949590200E+03   8.66028848648272500E+03
   2.30461334935895600E+04   2.09372081176427900E+04   2.09581453257604500E+04
  1.00000000
  0.00000000
C TL3
-1XX0004XX0006                2.  0.00              -2 1
      18       3.43904357882401600000E+02
   2.75748917329525200E+02   2.09556456539013700E+03  -3.02120107577198200E+02
   1.32522038307429600E+02   5.34206778326990500E+02   7.77786264529477200E+02
   1.49408699070299500E+03   7.42950753414815600E+03   3.11676509575282100E+04
   1.27236036995146400E+05   5.08023732184690700E+05   2.32321224243896200E+06
   5.09434464995741600E+06   3.65432987658662500E+06   2.66960449636756200E+06
   2.78771199529035400E+06   2.85557150987219100E+06   3.28340044978634400E+06
   1.55157367465512800E+00   3.13344553898543100E+00   3.37321729335057800E+00
   4.11467825483795900E+00   8.24190385834596400E+00   1.44386945071827500E+01
   4.50761403850135000E+01   3.97846218782673400E+02   1.72512452765138800E+03
   7.33827995192348500E+03   3.07002204634114500E+04   1.46775367026548200E+05
   6.61870409143599800E+05   1.37066806499039800E+06   1.35751400884485100E+06
   3.13588034329753700E+06   2.94525940129055000E+06   3.70207919969885800E+06
      12       7.75205848888472270000E-04
   5.16575344691927400E-02   1.53232837264703700E+00   2.37724925263830500E+00
   3.30486779211416800E+00   1.36818812823423600E+01   1.62481417148063300E+02
  -8.80925175507769900E+02   2.64168245174528800E+03   1.05532018711274100E+04
   4.46227461788609200E+06   1.84254666467924900E+08  -1.88729438474360000E+08
   9.86927346002103800E+00   2.75447572089606500E+02   4.20801419765296900E+02
   5.40238212815512500E+02   6.41977109380031700E+02   2.73672970771993400E+03
   6.16757885803951600E+03   5.72202239134570900E+03   1.66032197737507600E+04
   4.27963575320475200E+04   4.10830083290002900E+04   4.11240913373293300E+04
  1.00000000
  0.00000000
C TL4
-1XX0006XX0012                2.  0.00              -2 1
      18       3.43904357882401600000E+02
   2.75748917329525200E+02   2.09556456539013700E+03  -3.02120107577198200E+02
   1.32522038307429600E+02   5.34206778326990500E+02   7.77786264529477200E+02
   1.49408699070299500E+03   7.42950753414815600E+03   3.11676509575282100E+04
   1.27236036995146400E+05   5.08023732184690700E+05   2.32321224243896200E+06
   5.09434464995741600E+06   3.65432987658662500E+06   2.66960449636756200E+06
   2.78771199529035400E+06   2.85557150987219100E+06   3.28340044978634400E+06
   1.55157367465512800E+00   3.13344553898543100E+00   3.37321729335057800E+00
   4.11467825483795900E+00   8.24190385834596400E+00   1.44386945071827500E+01
   4.50761403850135000E+01   3.97846218782673400E+02   1.72512452765138800E+03
   7.33827995192348500E+03   3.07002204634114500E+04   1.46775367026548200E+05
   6.61870409143599800E+05   1.37066806499039800E+06   1.35751400884485100E+06
   3.13588034329753700E+06   2.94525940129055000E+06   3.70207919969885800E+06
      27       2.75716715953539190000E-04
   7.41855023793343800E-03   6.89364462199310900E-02   3.25985424621389200E-01
   4.37349506980829100E-01   5.33890733252180900E-01   6.63903346282451200E-01
   7.27352105811201600E-01   9.03208844084790000E-01   9.50187765078895200E-01
   1.25245027938036600E+00   3.52881270255085800E+00   1.26610600550988400E+01
   6.21970156366083600E+01   3.04227613082559300E+02   6.05579666456830200E+02
   1.13692588827181600E+03   2.18134235607163100E+03   2.55204251163790000E+03
   1.17699737622563400E+04  -6.30737653062193900E+04   9.35305340262357300E+04
   1.80726010396839700E+06  -1.83885262811079000E+06   8.69613895477062600E+05
  -8.57336616619472200E+05   8.19093664541233600E+05  -8.48869537345589800E+05
   5.83002815611244100E+00   5.40430558878495500E+01   2.51143324984160300E+02
   3.34941363004325500E+02   4.12846528154363100E+02   5.04528748011672400E+02
   5.79140621318636800E+02   6.59481393962392100E+02   7.64831178018867800E+02
   9.32370343345793200E+02   1.28049005297752600E+03   2.13574579051440000E+03
   2.76751077409838300E+03   6.01981074582288400E+03   1.05351913699554100E+04
   1.53693948316460300E+04   1.91191202315414500E+04   2.54555964205854500E+04
   4.10060803429461300E+04   9.20941067880203900E+04   7.30370500245080900E+04
   1.47396919251189900E+05   1.47544316170440800E+05   3.67181057879015600E+05
   3.67548238936894000E+05   1.81744201094532400E+05   1.81925945295626900E+05
  1.00000000
  0.00000000
/SWITCH
```

```
C < n 1>< n 2>< Tclose ><Top/Tde ><   Ie    ><Vf/CLOP ><  type  >
  XX0015XX0017      .05       .5                                        1
  XX0017A____1                                  MEASURING             1
/SOURCE
C < n 1><>< Ampl.  >< Freq.  ><Phase/T0><   A1   ><   T1   >< TSTART >< TSTOP  >
C Ideal voltage source
14XX0015 0       10.       60.                                    -1.        1.
/OUTPUT
BLANK BRANCH
BLANK SWITCH
BLANK SOURCE
BLANK OUTPUT
BLANK PLOT
BEGIN NEW DATA CASE
BLANK
```

A.4 ATP Data File for Robust TLNE Model

```
BEGIN NEW DATA CASE
C -----------------------------------------------------------
C Xin Nie
C Power Engineering Group
C Dept. of Electrical and Computer Engineering
C University of Alberta
C June 20, 2004
C -----------------------------------------------------------
POWER FREQUENCY                    60.
$DUMMY, XYZ000
C  dT  >< Tmax >< Xopt >< Copt >
   1.E-5      .2
     100       1       0       0       0       0       0       1       0
C        1       2       3       4       5       6       7       8
C [illegible]*90123456789012345678901234567890123456789012345678901234567890
/BRANCH
C Surface layer
C TL1 (reduced-order, after CLLSQ optimization)
-1IN___1A____1                2.  0.00              -2 1
       3       3.864633776250925100E+002
   1.314702435456360 1E+005   1.113044259871725 8E+004   7.146643589811817 5E+002
   3.087007541852243 8E+003   9.400320958020543 6E+000   2.679942459277918 9E+000
       1       1.022400031718859000E-003
   6.117460669092771 2E+003
   6.161905839764758 6E+003
  1.00000000
  0.00000000
C Deep region
C Low-order FDNE model
C BEGIN FDNE
$VINTAGE,1
C <BUS1><BUS2><BUS3><BUS4><   OHM        ><   milliH    ><  microF    >
C
C (1,1)
  A____1                    3.283409e+002
  A____1                    5.773775e+001   1.814903e+003
  A____1                    2.270264e+002   3.735426e+003
  A____1                    9.378893e+002   3.250459e+003
  A____1                    6.449830e+003   7.655092e+002
  A____1                   -3.774552e+002  -6.912906e+001
  A____1A 6__1              1.923339e+002   5.019495e+002
  A 6__1                   -1.770025e+004
  A 6__1                                                  8.039627e-001
  A____1A 8__1              2.399467e+002   1.253622e+003
  A 8__1                    3.599651e+004
  A 8__1                                                  1.305689e-001
  A____1A A__1              1.171002e+003   6.912165e+002
  A A__1                   -8.148504e+003
  A A__1                                                  9.737904e-002
  A____1A C__1              2.611214e+003   1.069758e+003
  A C__1                   -2.018949e+004
  A C__1                                                  2.721165e-002
$VINTAGE,0
C END FDNE
/SWITCH
C < n 1>< n 2>< Tclose ><Top/Tde ><   Ie    ><Vf/CLOP ><  type  >
  XX0015IN___1      .05       .5                                        1
/SOURCE
C < n 1><>< Ampl.  >< Freq.  ><Phase/T0><   A1   ><   T1   >< TSTART >< TSTOP  >
14XX0015 0       10.       60.                                    -1.        1.
/OUTPUT
BLANK BRANCH
BLANK SWITCH
BLANK SOURCE
BLANK INITIAL
BLANK OUTPUT
BLANK PLOT
BEGIN NEW DATA CASE
BLANK
```

B
Example 2

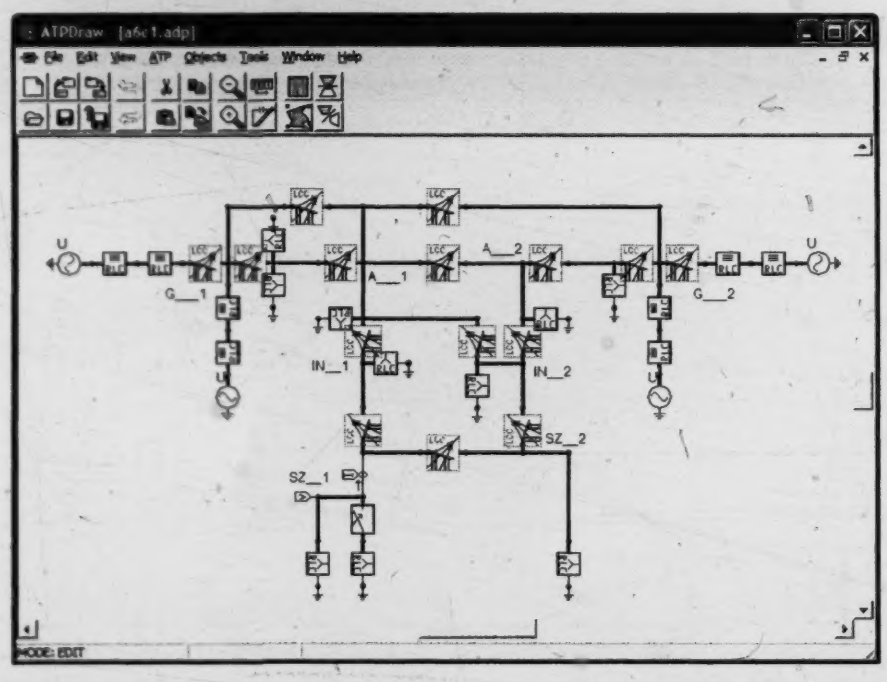
B.1 System Parameters

Elements	Parameters
TLS1	120km transmission line, one Hawk conductor per phase
TLS2	136km transmission line, one Drake conductor per phase
TLS3	165km transmission line, one Hawk conductor per phase
TL1	150km transmission line, one Drake conductor per phase
TL2	120km transmission line, one Drake conductor per phase
TL3	400km transmission line, one Hawk conductor per phase
TL4	220km transmission line, one Hawk conductor per phase
TL5	35km transmission line, one Hawk conductor per phase
TL6	10km transmission line, one Hawk conductor per phase
TL7	35km transmission line, one Hawk conductor per phase
TL8	15km transmission line, one Hawk conductor per phase
TL9	65km transmission line, one Hawk conductor per phase
TL10	133km transmission line, one Hawk conductor per phase
TL11	42km transmission line, one Hawk conductor per phase
TL12	375km transmission line, one Hawk conductor per phase
Tower	Conductors: horizontal -6, 0, 6m, vertical 15, 24, 15m; ground wires: horizontal -3.932, 3.932m, vertical 30, 30m, ground resistivity: 100Ωm
Load1	*RL* Load: 1200Ω, 500mH per phase
Load2	*RL* Load: 2150Ω, 380mH per phase
Load3	*RL* Load: 250Ω, 25mH per phase

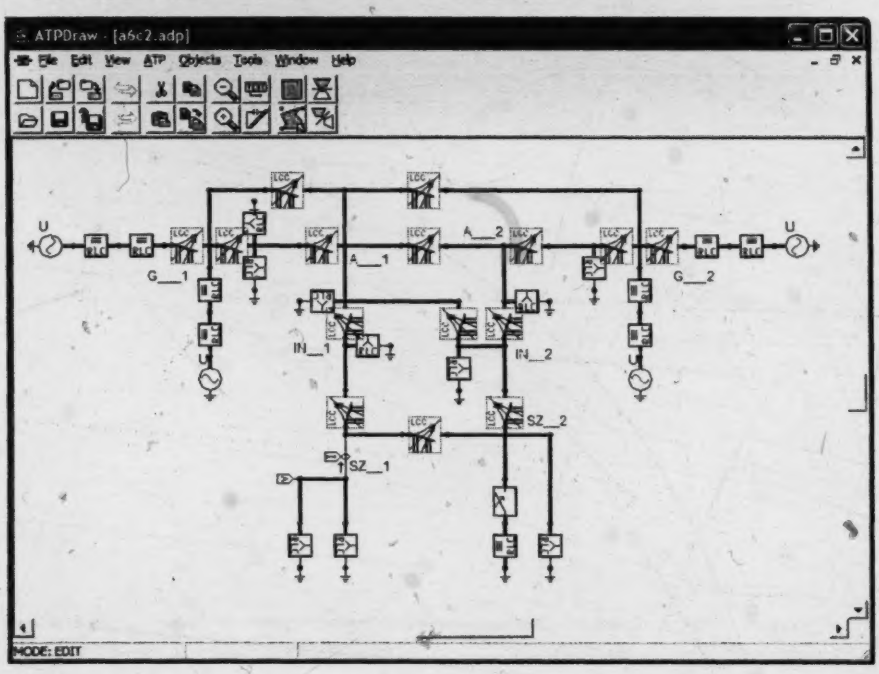
Load4	RL Load: 350Ω, 60mH per phase
Load5	RL Load: 250Ω, 25mH per phase
Load6	RL Load: 420Ω, 30mH per phase
Load7	RL Load: 200Ω, 130mH per phase
Load8	RL Load: 650Ω, 250mH per Phase
C1	Capacitor bank 5μF per phase
C2	Capacitor bank 20μF per phase
G1	Generator: 1.03$\angle$20.2°, Z_{G1}: 1.2Ω, 38.98mH per phase
G2	Generator: 1.01$\angle$10.5°, Z_{G2}: 1.1Ω, 45.52mH per phase
G3	Generator: 1.03$\angle$-6.8°, Z_{G3}: 0.9Ω, 38.98mH per phase
G4	Generator: 1.01$\angle$-17.0°, Z_{G4}: 0.8Ω, 35.23mH per phase
T1	Transformer: Z_{T1}: 1.5Ω, 23.4mH per phase
T2	Transformer: Z_{T2}: 0.8Ω, 29.5mH per phase
T3	Transformer: Z_{T3}: 1.6Ω, 23.4mH per phase
T4	Transformer: Z_{T4}: 0.6Ω, 20.8mH per phase

B.2 System Diagram in ATPDraw

B.2.1 Capacitor $C1$ Switching Case



B.2.2 Balanced Three-Phase to Ground Fault Case



B.3 ATP Data File for Full Model

B.3.1 Capacitor $C1$ Switching Case

```
BEGIN NEW DATA CASE
C ------------------------------------------------------------
C Xin Nie
C Power Engineering Group
C Dept. of Electrical and Computer Engineering
C University of Alberta
C September 10, 2004
C ------------------------------------------------------------
POWER FREQUENCY                    60.
$DUMMY, XYZ000
C  dT  >< Tmax >< Xopt >< Copt >
   2.E-5     .15
        1       1       0       0       0       0       0       1       0
C        1       2       3       4       5       6       7       8
C 345678901234567890123456789012345678901234567890123456789012345678901234567890
/BRANCH
C < n 1>< n 2><ref1><ref2>< R  >< L  >< C  >
C < n 1>< n 2><ref1><ref2>< R  >< A  >< B  ><Leng><><>0
  SZ__1A                  200.  130.                                          0
  X0007AX0009A              1.1 45.52                                         0
  X0010AX0012A              1.2 38.98                                         0
  X0009AX0046A               .8  29.5                                         0
  X0012AG___1A              1.5  23.4                                         0
  X0017A                                 8.                                  0
  SZ__2A                  650.  250.                                          0
  X0030AX0032A               .8 35.23                                         0
  X0032AX0049A               .6  20.8                                         0
  X0036AX0035A               .9 38.98                                         0
  G___2AX0036A              1.6  23.4                                         0
  X0042A                  420.   30.                                          0
  X0044A                  250.   25.                                          0
  X0044A                                20.                                   0
  IN__1A                 1200.  500.                                          0
  IN__2A                 2150.  380.                                          0
  A___1A                  250.   25.                                          0
  A___2A                  350.   60.                                          0
C TLS1 HAWK conductor
-1SZTMPAIN__1A                2.  0.00              -2
     12      4.090347064905760000E+02
  2.458905136002357000E+03   1.006919164736511000E+03   3.735522939590348000E+03
  4.949108642839249000E+03   1.809220174535260000E+03   1.626389975440227000E+03
  7.235902112013972000E+02   9.980615716539589000E+02   5.162740507080648000E+02
  1.179555959812405300E+04   6.126537567484807000E+04   2.428839145698563000E+06
  2.536132136180311000E+00   3.676274032523600000E+00   7.358252873864169000E+00
  1.840262484773079000E+01   3.580228449323820000E+01   5.945998927028807000E+01
  7.782171028008578000E+01   1.297276129271710000E+02   2.268054645514569000E+02
  4.144249177986411000E+03   2.155807371752076000E+04   8.578379257482028000E+05
     19      4.005194444611750000E-04
  1.569769744883085000E-02   5.969318016973556000E-02   7.657106329819022000E-02
  1.230253109782413000E-01   7.743626181762305000E+00   1.156313267927832000E+01
  1.517933952533545000E+01   2.454600998092000000E+01   2.175625245662915000E+02
  4.751575573024700000E+03   7.172772316756964000E+03   2.154208551130008000E+02
  1.927804409250900000E+05   1.001302044569728000E+05   6.676941419459103000E+05
  4.498508068621796000E+09  -4.592203111952144000E+09   3.249801162280406000E+09
 -3.157079140375575000E+09
  2.460067998651540000E+00   2.063829992047784000E+01   2.694103636262369000E+01
  4.366175083258754000E+01   2.728891534101922000E+03   3.969698516654513000E+03
  5.156233595789232000E+03   1.038658273690606000E+04   9.267324595579616000E+03
  1.124694586165610000E+05   6.932342002009935000E+04   4.989208395223323000E+05
  4.157544530856369000E+05   7.764223805854834000E+05   1.439408374462287000E+06
  4.025219577684497000E+06   4.029244797262184000E-06   4.381295402519039000E+06
  4.385676697921551000E+06
  1.00000000
  0.00000000
C TLS2 DRAKE conductor
-1SZTMPASZ__2A                2.  0.00              -2
     11      3.938973277035573000E+02
  2.271090787356759000E+03   2.468738369802207000E+03   2.767283308844692000E+03
  9.272013278576356000E+02   8.136523854897157000E+02   4.843864533476302000E+02
  6.006604547470770000E+02   3.191230068771898000E+02   1.474134983976601000E+04
  5.357097306002596000E+04   2.761989693560646000E+06
  2.565204294186215000E+00   6.238679152406237000E+00   1.373155019487035000E+01
  2.444115662193466000E+01   3.882345993654990000E+01   5.220646860706471000E+01
  8.949853843076818000E+01   1.612485445474848000E+02   6.100242815499842000E+03
  2.242497251791498000E+04   1.159710946925651000E+06
     19      4.539154979287958000E-04
  1.712759170053953000E-02   3.986913373903335000E-02   1.608642600299830000E-01
  6.890036182906788000E+00   8.786329257170928000E+00   1.187084000840551000E+01
  1.466504281283071000E+01   4.551321690537482000E+01   1.926819811653192000E+02
  2.213264955675185000E+03   1.110004761321304000E+04  -9.214328328701855000E+04
  2.817765442301621000E+05   1.678419416532637000E+04   1.225350369761673000E+06
 -3.168762404556547000E+07   2.934646328211559000E+07   3.778887191634843000E+07
 -3.689307291564467000E+07
  6.299201827904183000E+00   1.475354424616955000E+01   5.988920629316388000E+01
  2.468781047206587000E+03   3.246882256715642000E+03   4.205475680336974000E+03
  5.301662381272396000E+03   1.665669470442428000E+04   8.865719886677556000E+03
  4.657461284769958000E+04   1.133980748245041000E+05   4.435156084732816000E+05
  3.984178416006317000E+05   5.620686799638157000E+05   1.446828729109786000E+06
  2.871519673200630000E+06   2.874391192873833000E+06   3.654271889652527000E+06
  3.657926161542183000E+06
  1.00000000
  0.00000000
C TLS3 HAWK conductor
-1SZ__2AIN__2A                2.  0.00              -2
     12      4.090347064905760000E+02
  2.458905136002357000E+03   1.006919164736511000E+03   3.735522939590348000E+03
  4.949108642839249000E+03   1.809220174535260000E+03   1.626389975440227000E+03
  7.235902112013972000E+02   9.980615716539589000E+02   5.162740507080648000E+02
  1.179555959812405300E+04   6.126537567484807000E+04  -2.428839145698563000E+06
  2.536132136180311000E+00   3.678274032523600000E+00   7.358252873864169000E+00
  1.840262484773079000E+01   3.580228449323820000E+01   5.945998927028807000E+01
  7.782171028008578000E+01   1.297276129271710000E+02   2.268054645514569000E+02
  4.144249177986411000E+03   2.155807371752076000E+04   8.578379257482028000E+05
     19      5.505543845357618000E-04
  9.794656058558596000E-03   4.062127331237850000E-02   7.448549031790969000E-02
  4.002298701448773000E-02   6.562545649334067000E-02   7.793456900274408000E-02
  7.111514662870558000E-01   6.740052368561383000E+00   8.016207673581198000E+00
  1.172206700749057000E+01   3.502971302755199000E+01   7.054946753887513000E+01
  7.136004868170412000E+02   4.435453461485819000E+03   6.117812780315694000E+04
  6.793005712445018000E+04   7.497647220639265000E+05   2.364694593071054000E+07
 -2.453110096888204900E+07
  3.945877150681552000E+00   1.570359534061049000E+01   3.033936660197176000E+01
  1.670661222945046000E+01   2.608700546592988000E+01   3.324921072848005000E+01
  2.919886416106979000E+02   2.664618156511500000E+03   3.319650639220727000E+03
  4.566096983044453000E+03   7.473356722831026000E+03   6.699036297016883000E+03
  1.759025397917742000E+04   5.336861955214478000E+04   2.154763142183993000E+05
  3.138247760540839000E+05   9.375308613961636000E+05   1.823251480344845000E+06
  1.825074731825187000E+06
  1.00000000
  0.00000000
C TLI DRAKE conductor
-1IN__1AA___1A                2.  0.00              -2
     11      3.938973277035573000E+02
  2.271090787356759000E+03   2.468738369802207000E+03   2.767283308844692000E+03
  9.272013278576356000E+02   8.136523854897157000E+02   4.843864533476302000E+02
  6.006604547470770000E+02   3.191230068771898000E+02   1.474134983976601000E+04
  5.357097306002596000E+04   2.761989693560646000E+06
  2.565204294186215000E+00   6.238679152406237000E+00   1.373155019487035000E+01
  2.444115662193466000E+01   3.882345993654990000E+01   5.220646860706471000E+01
  8.949853843076818000E+01   1.612485445474848000E+02   6.100242815499842000E+03
  2.242497251791498000E+04   1.159710946925651000E+06
     18      5.005695258092510000E-04
  1.439029956896406000E-02   4.334580441565655000E-02   5.161860887968740E-02
  2.604730784090129000E+00   6.368875479882592000E+00   8.775205159527836000E+00
  1.077155715866400900E+01   1.624561224408722000E+01   1.492943549172489000E+02
  3.284343901369637000E+03   5.498535610569926000E+03   5.345331573082779000E+04
  2.315747160121790000E+05   3.272981976390529000E+05   3.141592781304394000E+08
 -3.105955744045123000E+08   9.307062210060334000E+08  -9.348912280105275000E+08
  5.619364715879287000E+00   1.683057604072452000E+01   2.052509261757305000E+01
```

```
  1.0172631145515070OE+03   2.5240302744971620OE+03   3.2850951560548700E+03
  4.1239859762206160OE+03   7.9000921196765590OE+03   7.1415901630257730OE+03
  8.6610718213955490OE+04   5.7671814537158200E+04   2.4351329338336270OE+05
  5.2641855170588490OE+05   9.5966080585034750OE+05   3.8727867307512700OE+06
  3.8766595174820250OE+06   2.7539784907934830OE+06   2.7567324692842790OE+06
 1.00000000
 0.00000000
C TL2 DRAKE conductor
-1IN__2AA___2A                2.  0.00                -2
     11       3.9389732770355738000E+02
  2.2710907873567590OE+03   2.4687383698022070OE+03   2.7672833088446920OE+03
  9.2720132785635600E+02   8.1365238548971570OE+02   4.8438645334763020OE+02
  6.0066045474707700E+02   3.1912300687718980OE+02   1.4741349839766010OE+04
  5.3570973060025960E+04   2.7619896935064600E+06
  2.5652042941862150OE+00   6.2386791524062370OE+00   1.3731550194870350OE+01
  2.4441156621934660OE+01   3.8823459936549900E+01   5.2204648607064710OE+01
  8.9498538430768180OE+01   1.6124854454748480OE+02   6.1002428154998420OE+03
  2.2424972517914980OE+04   1.1597109469256140E+06
     20       4.0031742713365390000E-04
  2.0889143039510750OE-02   4.3259348901814790OE-02   1.1525151756945020OE+00
  8.8498154268285860E+00   1.0644495433243630OE+01   1.3976609684085440OE+01
  1.9136348721895170OE+01   6.0331418366612060OE+01   3.0497627949848180OE+02
  2.5613574005403300OE+03   1.6508359804979780OE+04   5.7065070206066260OE+04
 -1.5311268222313170OE+05   3.1699898467954310OE+05   4.9248446818344490OE+04
  1.1900845760824020OE+06   1.6835002828101870OE+08  -1.7220292555184890OE+08
  1.1673463368986210OE+08  -1.1436150966343520OE+08
  7.4619747594449200E+00   1.5574858448693060OE+01   4.1523192753806310OE+02
  3.0783555515815590OE+03   3.8858322501365650OE+03   4.8558793197102950OE+03
  6.7333441172706900E+03   2.1469926317737040OE+04   1.3430028894747300OE+04
  5.3360359957215260OE+04   1.5361572352437940OE+05   3.5473523393128840OE+05
  5.7523615170071960OE+05   5.8050759596729800OE+05   7.4589394516250720OE+05
  1.6823620430225500OE+06   3.6272826451807740OE+06   3.6309099278259490OE+06
  4.4806921768242080OE+06   4.4851728690010250OE+06
 1.00000000
 0.00000000
C TL3 HAWK conductor
-1IN__2AA___1A                2.  0.00                -2
     12       4.0903470649057660000E+02
  2.4589051360023570OE+03   1.0069191647365110OE+03   3.7355229395903480OE+03
  4.9491086428392490OE+03   1.8092201745352600OE+03   1.6263899754402270OE+03
  7.2359021120139720OE+02   9.9806157165395890OE+02   5.1627405070806480OE+02
  1.1795559581240530OE+04   6.1265375674848070OE+04   2.4288391456985630OE+06
  2.5361321361803110OE+00   3.6782740325236000E+00   7.3582528738641690OE+00
  1.8402624847730790OE+01   3.5802284493238200E+01   5.9459989270288070OE+01
  7.7821710280008570OE+01   1.2972761292717100E+02   2.2680546455145690OE+02
  4.1442491779864110OE+03   2.1558073717520760OE+04   8.5783792574820280OE+05
     30       1.3364385475383520000E-03
  9.6897671350396270OE-03   1.3312020476810650OE-02   2.1856842721849400OE-02
  2.1967556424880930OE-02   3.3807808519096020OE-02   4.3348677104904930OE-02
  6.7610624695776440OE-02   5.4314041178945980OE-02   1.4146038979994510OE-01
  9.1100215767050520OE-02   8.0118098354694100E-02   1.1245333455963980OE-01
  1.3807669291973040OE-01   1.8246365962366120OE-01   2.6409961844753850OE+00
  3.8705219695051040OE+00   6.1293882773385870OE+00   1.9099381039167440OE+01
  7.9653824316134180OE+01   1.2905664136843410OE+02   9.0165084060503090OE+02
  7.9997382731351950OE+03   4.4525546914192240OE+03   1.4227043478709970OE+04
  1.5084440311967440OE+04   4.4837532864447000OE+05  -8.3270088597577960OE+07
  8.2655971697383340OE+07   6.5944283513351500E+06  -6.4751936781113860OE+06
  3.7263583768999040OE+00   5.1689417453996760OE+00   8.2381736152278030OE+00
  8.8439065547048690OE+00   1.3275561914201370OE+01   1.6874891312812570OE+01
  2.3264377807936730OE+01   2.3767549562594390OE+01   2.8310737437085220OE+01
  3.4615868427781760OE+01   3.5741880447729280OE+01   4.5348204773341930OE+01
  5.7726202591238030OE+01   7.5940415502731870OE+01   1.0488165819029400OE+03
  1.6447899013116057000E+03   2.3664638573296890OE+03   3.5601148774115630OE+03
  5.8398243420261820OE+03   6.4207286909255900OE+03   1.1389217416035240OE+04
  5.3551145696296040OE+04   9.5609367636291370OE+04   7.2964650275121750OE+04
  1.3177606065573120OE+05   3.4656937806718900E+05   6.0329056656339200E+05
  6.0389386023702600E+05   7.5063523809546370OE+05   7.5138587333557900E+05
 1.00000000
 0.00000000
C TL4 HAWK conductor
-1A__1AA___2A                 2.  0.00                -2
     12       4.0903470649057660000E+02
  2.4589051360023570OE+03   1.0069191647365110OE+03   3.7355229395903480OE+03
  4.9491086428392490OE+03   1.8092201745352600OE+03   1.6263899754402270OE+03
  7.2359021120139720OE+02   9.9806157165395890OE+02   5.1627405070806480OE+02
  1.1795559581240530OE+04   6.1265375674848070OE+04   2.4288391456985630OE+06
  2.5361321361803110OE+00   3.6782740325236000E+00   7.3582528738641690OE+00
  1.8402624847730790OE+01   3.5802284493238200E+01   5.9459989270288070OE+01
  7.7821710280008570OE+01   1.2972761292717100E+02   2.2680546455145690OE+02
  4.1442491779864110OE+03   2.1558073717520760OE+04   8.5783792574820280OE+05
     18       3.3358038754720500000E-05
  6.5887722409255740OE+00   9.2113221746047390OE+01   1.6423682191281080OE+02
  3.0178407403080660OE+02   4.0728514877113830OE+02   7.2020971549401800OE+02
  2.1160068828669650OE+03   9.5441739377620700OE+03  -7.2228283561231840OE+05
  1.0059208008146480OE+06   1.9061242592731810OE+06   1.8683249264185610OE+06
  1.2034035732374430OE+07   3.9370274836321700OE+07  -3.2956202374353530OE+07
  1.1817207962102770OE+08   9.0354875134119520OE+10  -9.0495566761482650OE+10
  2.3371251872899270OE+03   3.2411038576230860OE+04   5.8069296864557620OE+04
  1.0495280561240230OE+05   1.4968018477432050OE+05   2.3961771850323940OE+05
  3.3704846393224670OE+05   4.2688804483029970OE+05   2.1825853821176600OE+06
  2.1590159257941460OE+06   9.3528383821623300OE+06   1.5716003127905200OE+07
  4.2748384979229000OE+07   6.3175356915514250OE+07   6.8459854134571280OE+07
  4.8514755209822910OE+08   2.9449956354442660OE+08   2.9479406310797130OE+08
 1.00000000
```

```
  0.00000000
C TL7 HAWK conductor
-1A___2AX0042A                    2.  0.00                -2
     12        4.09034706490576600000E+02
  2.45890513600235700E+03   1.00691916473651100E+03   3.73552293959034800E+03
  4.94910864283924900E+03   1.80922017453526000E+03   1.62638997544022700E+03
  7.23590211201397200E+02   9.98061571653958900E+02   5.16274050708064800E+02
  1.17955595981240530E+04   6.12653756748480700E+04   2.42883914569856300E+06
  2.53613213618031100E+00   3.67827403252360000E+00   7.35825287386416900E+00
  1.84026248477307900E+01   3.58022844932382000E+01   5.94599892702880700E+01
  7.78217102800857800E+01   1.29727612927171000E+02   2.26805464551456900E+02
  4.14424917798641100E+03   2.15580737175207600E+04   8.57837925748202800E+05
     24        1.16743539295552900000E-04
  1.40338144245651000E-02   3.39703490032430100E-02   7.69056681810012900E+00
  3.58110106599894800E+01   4.59434851394437100E+01   2.84917356432648400E+02
  3.53388521104483100E+01  -2.01507841110737700E+02   6.39382485292421200E+01
  8.88535598132061800E+01   3.01026729828361200E+03   1.26069665984102700E+04
  6.12738155584614600E+04   2.34189902042343900E+05   1.31935599094607520OE+05
  4.94675134450245700E+05   5.84566299761196800E+05   2.68740374944970800E+06
 -2.54293806583983400E+05   2.05396861193992900E+06   2.67442299542458200E+07
 -2.64255488222761200E+07   1.13743438258904500E+09  -1.14376276468576100E+09
  9.70708063378627600E+00   2.35778474391517000E+01   2.65822018541962300E+03
  1.20880840340937600E+04   1.54883460011065300E+04   1.90230929114263600E+04
  2.53178115338652200E+04   1.90298436673898000E+04   2.24482361683339700E+04
  3.11892517616459600E+04   1.21614891415593700E+05   2.55166226779757100E+05
  6.08626632094715300E+05   1.50779229197664000E+06   1.83794319683241000E+06
  2.82950218682721500E+06   4.40043917681811400E+06   7.89472474983174600E+06
  1.11535384043682500E+07   1.93763328756605500E+07   2.25132290200842800E+08
  2.25357422491043800E+08   3.24932297557221100E+07   3.25257229854777800E+07
  1.00000000
  0.00000000
C TL8 HAWK conductor
-1X0049AG___2A                    2.  0.00                -2
     12        4.09034706490576600000E+02
  2.45890513600235700E+03   1.00691916473651100E+03   3.73552293959034800E+03
  4.94910864283924900E+03   1.80922017453526000E+03   1.62638997544022700E+03
  7.23590211201397200E+02   9.98061571653958900E+02   5.16274050708064800E+02
  1.17955595981240530E+04   6.12653756748480700E+04   2.42883914569856300E+06
  2.53613213618031100E+00   3.67827403252360000E+00   7.35825287386416900E+00
  1.84026248477307900E+01   3.58022844932382000E+01   5.94599892702880700E+01
  7.78217102800857800E+01   1.29727612927171000E+02   2.26805464551456900E+02
  4.14424917798641100E+03   2.15580737175207600E+04   8.57837925748202800E+05
     12        5.00031399666535300000E-05
  2.20065054666852400E+02   1.99171330798515600E+01   3.03245876828597800E+02
  1.33319469316572800E+03   6.10656214763100300E+03   4.90731603290800800E+04
  8.24204645125399000E+04   1.72902848804883900E+06   1.92561728913267900E+06
  8.06335734716742400E+06   1.95628203187250900E+10  -1.95746777984592000E+10
  4.13655643066896300E+04   3.81062807774905700E+03   5.91446709330260600E+04
  2.25671186581180700E+05   2.86301298194201800E+05   9.01323527485587800E+05
  1.16379350765599500E+06   1.36017371159249100E+07   5.83800344864872500E+06
  2.64279686980491800E+07   1.11483362311402800E+08   1.11594845673714300E+08
  1.00000000
  0.00000000
C TL9 HAWK conductor
-1X0046AX0044A                    2.  0.00                -2
     12        4.09034706490576600000E+02
  2.45890513600235700E+03   1.00691916473651100E+03   3.73552293959034800E+03
  4.94910864283924900E+03   1.80922017453526000E+03   1.62638997544022700E+03
  7.23590211201397200E+02   9.98061571653958900E+02   5.16274050708064800E+02
  1.17955595981240530E+04   6.12653756748480700E+04   2.42883914569856300E+06
  2.53613213618031100E+00   3.67827403252360000E+00   7.35825287386416900E+00
  1.84026248477307900E+01   3.58022844932382000E+01   5.94599892702880700E+01
  7.78217102800857800E+01   1.29727612927171000E+02   2.26805464551456900E+02
  4.14424917798641100E+03   2.15580737175207600E+04   8.57837925748202800E+05
     19        2.16779123239017300000E-04
  2.56688120191699100E-02   7.45735865425355800E-02   2.61582425335350600E+00
  1.58214000348970100E+01   2.01577916215819100E+01   2.30952255581425600E+01
  3.75136258999867100E+01   9.54682829266683500E+01   6.04125675569373400E+02
  9.76486155231744300E+03   1.47064885595657300E+04   3.54253022157960700E+05
  3.59232206937071600E+04   2.77681421428246600E+05   7.81674620583545300E+05
  7.74712244871832800E+09  -7.59800697692648700E+09   1.35513236224946500E+10
 -1.37019138968198200E+10
  9.27896860203729900E+00   2.71072079148495600E+01   5.51044370695290500E+02
  5.58492220807026200E+03   7.29437451310514800E+03   8.31674043053815300E+03
  1.33367902521258300E+04   3.60996353432424400E+04   2.68346709221327600E+04
  1.54366136678135800E+05   1.93572363341506500E+05   8.84204829000300700E+05
  1.08374170276550300E+06   1.76932522625841200E+06   2.79314179316203100E+06
  1.24037983078214100E+07   1.24162021061292100E+07   1.07871612508418900E+07
  1.07979484120927400E+07
  1.00000000
  0.00000000
C TL10 HAWK conductor
-1X0046AA___1A                    2.  0.00                -2
     12        4.09034706490576600000E+02
  2.45890513600235700E+03   1.00691916473651100E+03   3.73552293959034800E+03
  4.94910864283924900E+03   1.80922017453526000E+03   1.62638997544022700E+03
  7.23590211201397200E+02   9.98061571653958900E+02   5.16274050708064800E+02
  1.17955595981240530E+04   6.12653756748480700E+04   2.42883914569856300E+06
  2.53613213618031100E+00   3.67827403252360000E+00   7.35825287386416900E+00
  1.84026248477307900E+01   3.58022844932382000E+01   5.94599892702880700E+01
  7.78217102800857800E+01   1.29727612927171000E+02   2.26805464551456900E+02
  4.14424917798641100E+03   2.15580737175207600E+04   8.57837925748202800E+05
     19        4.43851439830561100000E-04
  1.38467332808500700E-02   6.36120257233293700E-02   4.20656661892195000E-02
  8.42940619988564900E-02   2.78303245931058400E-01   8.63394200389837200E+00
  1.04168457038902800E+01   2.26407709951103000E+01   1.63224210175601300E+02
  5.17641398064459300E+03   4.74610120033646900E+03   2.66323812290835400E+04
  1.28306093094039200E+05   4.41488616776506300E+04   1.15326362089665700E+06
 -8.28703063596845900E+07   8.15012011146409800E+07   9.07248589988122100E+03
 -2.44611081418986700E+03
  5.05436302187456200E+00   1.92284514288763800E+01   1.94815172706510500E+01
  3.13830071121717000E+01   1.03833464543928400E+02   3.08412954244814100E+03
  3.88989447310648000E+03   8.89831389746010600E+03   7.36254554745701100E+03
  1.10802992274832600E+05   5.06775394330296000E+04   4.12837881301153100E+05
  3.30469476516399400E+05   5.71943555824721700E+05   1.39415645997757700E+06
  2.64047008033511200E+06   2.64311055041544700E+06   3.73512434447443600E+06
  3.73885946881890500E+06
  1.00000000
  0.00000000
C TL11 HAWK conductor
-1X0042AX0049A                    2.  0.00                -2
     12        4.09034706490576600000E+02
  2.45890513600235700E+03   1.00691916473651100E+03   3.73552293959034800E+03
  4.94910864283924900E+03   1.80922017453526000E+03   1.62638997544022700E+03
  7.23590211201397200E+02   9.98061571653958900E+02   5.16274050708064800E+02
  1.17955595981240530E+04   6.12653756748480700E+04   2.42883914569856300E+06
  2.53613213618031100E+00   3.67827403252360000E+00   7.35825287386416900E+00
  1.84026248477307900E+01   3.58022844932382000E+01   5.94599892702880700E+01
  7.78217102800857800E+01   1.29727612927171000E+02   2.26805464551456900E+02
  4.14424917798641100E+03   2.15580737175207600E+04   8.57837925748202800E+05
     21        1.40129890129590700000E-04
  2.82409465040976800E+02   5.73666755142106500E-01   2.17536291440966400E+01
  3.46969988477248620E+01   3.86322638506245000E+01   2.72358421123721800E+01
  8.30384774679833600E+01   5.44278581191543800E+01  -7.83622740098267300E+02
  8.53611012878148500E+03   4.52847893308504100E+04   2.09702191470714500E+05
  7.34012430467903700E+05   9.48685976801616600E+04   1.09528829765371700E+06
  1.27602110708312400E+06   4.46680320404957500E+05   8.92491202129325800E+06
 -8.81159104863197700E+06   1.10155759833908800E+09  -1.10558172992019600E+09
  1.05642363611503200E+01   2.15117728119521300E+02   8.00177968562786300E+03
  1.17453901961811200E+04   1.25836048886094200E+04   9.82315069880778500E+03
  1.22450233355681700E+04   2.04493234107368700E+04   1.35561763828908700E+05
  1.31015066778640600E+05   4.65519105469535800E+05   1.07289979236858500E+06
  2.39840255363253400E+06   2.93078299179311300E+06   4.87134948866919700E+06
  8.30249729851572600E+06   1.23018882615654100E+07   2.64849913464487000E+08
  2.65114763377951000E+08   2.15065384249761900E+07   2.15280449634011800E+07
  1.00000000
  0.00000000
C TL12 HAWK conductor
-1A___1AX0049A                    2.  0.00                -2
     12        4.09034706490576600000E+02
  2.45890513600235700E+03   1.00691916473651100E+03   3.73552293959034800E+03
  4.94910864283924900E+03   1.80922017453526000E+03   1.62638997544022700E+03
  7.23590211201397200E+02   9.98061571653958900E+02   5.16274050708064800E+02
  1.17955595981240530E+04   6.12653756748480700E+04   2.42883914569856300E+06
  2.53613213618031100E+00   3.67827403252360000E+00   7.35825287386416900E+00
  1.84026248477307900E+01   3.58022844932382000E+01   5.94599892702880700E+01
```

```
  7.7821710280008578O0E+01   1.2972761292771000E+02   2.2680546455145690OE+02
  4.1442491779864110OE+03   2.1558073717520760OE+04   8.5783792574820280OE+05
    30          1.2528342737275560000E-03
  9.0875731152902480OE-03   1.2492196540588440OE-02   2.0468401301447800OE-02
  2.0661532739769050OE-02   3.3175982394844250OE-02   4.2017652418513750OE-02
  4.2378790661278540OE-02   1.0187900731190330OE-01   7.7839404651275010OE-02
 -2.0234955598607350OE-01   3.1037511585299100OE-01   7.8621512967940370OE-02
  1.0597860940406800OE-01   1.2726767061255880OE-01   1.7199869490567120OE-01
  2.7996466839156440OE+00   3.6338049920895680OE+00   5.7444286922030130OE+00
  1.7859778099094200OE+01   7.0375803865122420OE+01   1.3328731665197840OE+02
  8.0871102308655280OE+02   7.7112631920639980OE+03   1.7059793543507310OE+04
  1.5518581957275650OE+04   3.5051223761535730OE+05   9.5909087356939380OE+11
 -5.0148935968136420OE+12   5.0128017351239070OE+12  -9.5699940372488390OE+11
  3.7266626025894410OE+00   5.1693637458373960OE+00   8.2388446280463190OE+00
  8.8446302644929680OE+00   1.3796047943109550OE+01   1.8222581565829710OE+01
  1.7115660097284050OE+01   2.8797402483226990OE+01   3.3457801614307520OE+01
  2.9143519153674430OE+01   2.9128007690728020OE+01   3.5744795943836220OE+01
  4.5351989596039250OE+01   5.6633658869029890OE+01   7.5946586756261040OE+01
  1.1768866418162250OE+03   1.6449128760199310OE+03   2.3666547586126380OE+03
  3.5606885179091000OE+03   5.8417670197897670OE+03   6.6763076846881630OE+03
  1.0989091203861670OE+04   8.2756567322428310OE+04   6.2906098554807050OE+04
  1.3247314649267300E+05   3.4056499567374510OE+05   7.3714299774282040OE+05
  7.3788014074056390OE+05   7.3822866751377750OE+05   7.3896689618129180OE+05
  1.00000000
  0.00000000
/SWITCH
C < n 1>< n 2>< Tclose ><Top/Tde ><    Ie    ><Vf/CLOP ><  type  >
  X0017AGZ__1A        .05         1.                                          0
  SZ__1ASZTMPA                                      MEASURING               1
/SOURCE
C < n 1><>< Ampl.  >< Freq.  ><Phase/T0><   A1   ><   T1   >< TSTART >< TSTOP  >
14X0007A 0     195.       60.       10.5                                -1.      1.
14X0010A 0     200.       60.       20.2                                -1.      1.
14X0030A 0     195.       60.       -17.                                -1.      1.
14X0035A 0     200.       60.       -6.8                                -1.      1.
/OUTPUT
  SZ__1ASZ__2A
BLANK BRANCH
BLANK SWITCH
BLANK SOURCE
BLANK OUTPUT
BLANK PLOT
BEGIN NEW DATA CASE
BLANK
```

B.3.2 Balanced Three-Phase to Ground Fault Case

```
BEGIN NEW DATA CASE
C --------------------------------------------------------
C Xin Nie
C Power Engineering Group
C Dept. of Electrical and Computer Engineering
C University of Alberta
C September 22, 2004
C --------------------------------------------------------
POWER FREQUENCY                        60.
$DUMMY, XYZ000
C  dT  >< Tmax >< Xopt >< Copt >
   2.E-5      .2
       1       1       2       0       0       0       0       0       1       0
C      1       2       3       4       5       6       7       8
C 345678901234567890123456789012345678901234567890123456789012345678901234567890
/BRANCH
C < n 1>< n 2><refl1><ref2>< R  >< L  >< C  >
C < n 1>< n 2><refl1><ref2>< R  >< A  >< B  ><Leng><><>0
  SZ__1A                  200.  130.                                        0
  X0007AX0009A             1.1 45.52                                        0
  X0010AX0012A             1.2 38.98                                        0
  X0009AX0046A              .8  29.5                                        0
  X0012AG___1A             1.5  23.4                                        0
  SZ__1A                                  8.                                0
  SZ__2A                650.  290.                                          0
  X0030AX0032A           .8 35.23                                           0
  X0032AX0049A            .6  20.8                                          0
  X0035AX0036A           .9 38.98                                           0
  G___2AX0036A          1.6  23.4                                          0
  X0042A                420.   30.                                         0
  X0044A                250.   25.                                         0
  X0044A                                 20.                               0
  IN__1A               1200.  500.                                         0
  IN__2A               2150.  380.                                         0
  A__1A                 250.   25.                                         0
  A__2A                 350.   60.                                         0
       X0017A                2.
C TLS1 HAWK conductor
-1SZTMPAIN__1A                2.  0.00                -2
      12          4.0903470649057660000E+02
   2.4589051360O235700E+03   1.0069191647365110OE+03   3.7355229395903480OE+03
   4.9491086428392490OE+03   1.8092201745352600OE+03   1.6263899754402270OE+03
   7.2359021120139720OE+02   9.9806157165395890OE+02   5.1627405070806480OE+02
   1.1795559981240530OE+04   6.1265375674848070OE+04   2.4288391456985630OE+06
   2.5361321361803110OE+00   3.6782740325236000OE+00   7.3582528738641690OE+00
   1.8402624847730790OE+01   3.5802284493238200OE+01   5.9459989270288070OE+01
   7.7821710280008578OE+01   1.2972761292717100OE+02   2.2680546455145690OE+02
   4.1442491779864110OE+03   2.1558073717520760OE+04   8.5783792574820280OE+05
     19          4.0051944446117500000E-04
   1.5697697448830850OE-02   5.9693180169735560OE-02   7.6571063298190220OE-02
   1.2302531097824130OE-01   7.7436261817623050OE+00   1.1563132679278320OE+01
   1.5179339525354500E+01   2.4546009980920000E+01   2.1756252456629150OE+02
   4.7515755730247000E+03   7.1727723167569640OE+03   2.1542085511300080OE+02
   1.9278044092509000E+05   1.0013020445697280OE+05   6.6769414194591030OE+05
   4.4985080686217960OE+09  -4.5922031119521440OE+09   3.2498011622804060OE+09
  -3.1570791403755750OE+09
   5.4600679986515400OE+00   2.0638299920477840OE+01   2.6941036362623690OE+01
   4.3661750832587540OE+01   2.7288915341019220OE+03   3.9696985166545130OE+03
   5.1562335957892320OE+03   1.0386582736906060OE+04   9.2673245955796160OE+03
   1.1246945861656100OE+05   6.9323420020099350OE+04   4.9892083952233230OE+05
   4.1575445308563690OE+05   7.7642238058548340OE+05   1.4394483744622870OE+06
   4.0252195776844970OE+06   4.0292447972621840OE+06   4.3812954025190390OE+06
   4.3856766979215510OE+06
  1.00000000
  0.00000000
C TLS2 DRAKE conductor
-1SZTMPASZ__2A                2.  0.00                -2
      11          3.9389732770355730000E+02
   2.2710907873567590OE+03   2.4687383698022070OE+03   2.7672833088446920OE+03
   9.2720132785763560OE+02   8.1365238548971570OE+02   4.8438645334763020OE+02
   6.0066045474707700E+02   3.1912300687718980OE+02   1.4741349839766010OE+04
   5.3570973060025960OE+04   2.7619896935606460OE+06
   2.5652042941862150OE+00   6.2386791524062370OE+00   1.3731550194870350OE+01
   2.4441156621934660OE+01   3.8823459936549900OE+01   5.2204648607064710OE+01
   8.9498538430768180OE+01   1.6124854454748480OE+02   6.1002428154998420OE+03
   2.2424972517914980OE+04   1.1597109469256510OE+06
     19          4.5391549792879580000E-04
   1.7127591700539530OE-02   3.9869133739033350OE-02   1.6086426002998300OE-01
   6.8900361829067880OE+00   8.7863292571709280OE+00   1.1870840008405510OE+01
   1.4665042812830710OE+01   4.5513216905378820OE+01   1.9268198116531920OE+02
   2.2132649556751850OE+03   1.1193047613213040OE+04  -9.2143283287018550OE+04
   2.8177654423016220OE+05   1.6784194165326370OE+04   1.2253503697616730OE+06
  -3.1687624045565470OE+07   2.9346463282115590OE+07   3.7788871916348300OE+07
  -3.1689307291564467OE+07
   6.2992018279041830OE+00   1.4753544246169550OE+01   5.9889206293163880OE+01
   2.4687810472065870OE+03   3.2468822567156420OE+03   4.2054756803369740OE+03
   5.3016623812723960OE+03   1.6656694704424280OE+04   8.8657198866775560OE+03
   4.6574612847699580OE+04   1.1339807482450410OE+05   4.4351560847328160OE+05
   3.9841784160063170OE+05   5.6206867996381570OE+05   1.4468287291097860OE+06
   2.8715196732006300OE+06   2.8743911928738330OE+06   3.6542718896525270OE+06
   3.6579261615421830OE+06
  1.00000000
  0.00000000
C TLS3 HAWK conductor
-1SZ__2AIN__2A                2.  0.00                -2
      12          4.0903470649057660000E+02
```

```
  2.458905136002357 00E+03   1.006919164736511 00E+03   3.735522939590348 00E+03
  4.949108642839249 00E+03   1.809220174535260 00E+03   1.626389975440227 00E+03
  7.235902112013972 00E+02   9.980615716539589 00E+02   5.162740507080648 00E+02
  1.179555959812405 30E+04   6.126537567484807 00E+04   2.428839145698563 00E+06
  2.536132136180311 00E+00   3.678274032523600 00E+00   7.358252873864169 00E+00
  1.840262484773079 00E+01   3.580228449323820 00E+01   5.945599892702880 70E+01
  7.782171028000857 80E+01   1.297276129271710 00E+02   2.268054645514569 00E+02
  4.144249177996411 00E+03   2.155807371752076 00E+04   8.578379257482028 00E+05
     19          3.505543845357610 0000E-04
  9.794656058558596 00E-03   4.062127331237850 00E-02   7.448549031790969 00E-02
  4.002298701448773 00E-02   6.562541[illegible]33406700E-02   7.793456900274408 00E-02
  7.111514662870558 00E-01   6.740[illegible]68561383 00E+00   8.016207673581198 00E+00
  1.172206700749057 00E+01   3.502971302755199 00E+01   7.054946753887513 00E+01
  7.136004868170412 00E+02   4.435453461485819 00E+03   6.117812783015694 00E+04
  6.793005712445018 00E+04   7.497647220632265 00E+05   2.364694593071054 00E+07
 -2.453110096882049 00E+07
  3.945877150681552 00E+00   1.570359534061049 00E+01   3.033936660197176 00E+01
  1.670661222945046 00E+01   2.608700546592988 00E+01   3.324921072848005 00E+01
  2.919886416106979 00E+02   2.664618156511500 00E+03   3.319650639220727 00E+03
  4.566096983044453 00E+03   7.473356722831026 00E+03   6.699036297016883 00E+03
  1.759025397917742 00E+04   5.336861955214478 00E+04   2.154763142183993 00E+05
  3.138247760540839 00E+05   9.375308613961636 00E+05   1.823251480344845 00E+06
  1.825074731825187 00E+06
 1.00000000
 0.00000000
C TL1 DRAKE conductor
-1IN__1AA___1A                2.  0.00              -2
     11        3.938973277035573 0000E+02
  2.271090787356759 00E+03   2.468738369802207 00E+03   2.767283308844692 00E+03
  9.272013278576356 00E+02   8.136523854897157 00E+02   4.843864533476302 00E+02
  6.006604547470770 00E+02   3.191230068771898 00E+02   1.474134983976601 00E+04
  5.357097306002596 00E+04   2.761989693560646 00E+06
  2.565204294186215 00E+00   6.238679152406237 00E+00   1.373155019487035 00E+01
  2.444115662193466 00E+01   3.882345993654990 00E+01   5.220646860706471 00E+01
  8.949853843076818 00E+01   1.612485445474848 00E+02   6.100242815499842 00E+03
  2.242497251791498 00E+04   1.159710946925651 00E+06
     18        5.005695258092501 0000E-04
  1.439029956896406 00E-02   4.334580441565655 00E-02   5.161860887968794 00E-02
  2.604730784090129 00E+00   6.368875479882592 00E+00   8.775250159527836 00E+00
  1.077155715864409 00E+01   1.624561224408722 00E+01   1.492943549172489 00E+02
  3.284343901369637 00E+03   5.498535610569926 00E+03   5.345331573082779 00E+04
  2.315747160121790 00E+05   3.272981976390529 00E+05   3.141592781304394 00E+08
 -3.105955744045123 00E+08   9.307062210060334 00E+08  -9.348912280105275 00E+08
  5.619364715872870 00E+00   1.683057604072452 00E+01   2.052509261757305 00E+01
  1.017263114551507 00E+03   2.524030274497162 00E+03   3.285095156054870 00E+03
  4.123985976220616 00E+03   7.900092119676559 00E+03   7.141590163025773 00E+03
  8.661071821395549 00E+04   5.767181453715820 00E+04   2.435132933833627 00E+05
  5.264185517058849 00E+05   9.596608058503475 00E+05   3.872786730751270 00E+06
  3.876659517482025 00E+06   2.753978490793403 00E+06   2.756732469284279 00E+06
 1.00000000
 0.00000000
C TL2 DRAKE conductor
-1IN__2AA___2A                2.  0.00              -2
     11        3.938973277035573 0000E+02
  2.271090787356759 00E+03   2.468738369802207 00E+03   2.767283308844692 00E+03
  9.272013278576356 00E+02   8.136523854897157 00E+02   4.843864533476302 00E+02
  6.006604547470770 00E+02   3.191230068771898 00E+02   1.474134983976601 00E+04
  5.357097306002596 00E+04   2.761989693560646 00E+06
  2.565204294186215 00E+00   6.238679152406237 00E+00   1.373155019487035 00E+01
  2.444115662193466 00E+01   3.882345993654990 00E+01   5.220646860706471 00E+01
  8.949853843076818 00E+01   1.612485445474848 00E+02   6.100242815499842 00E+03
  2.242497251791498 00E+04   1.159710946925651 00E+06
     20        4.003174271336539 0000E-04
  2.088914303951075 00E-02   4.325934890181479 00E-02   1.152515175694502 00E+00
  8.849815426828586 00E+00   1.064449543324363 00E+01   1.397660968408544 00E+01
  1.913634872189517 00E+01   6.033141836661206 00E+01   3.049767294984818 00E+02
  2.561357400540433 00E+03   1.650835980497978 00E+04   5.706507020606626 00E+04
 -1.553112682223131 70E+05   3.169989846795431 00E+05   4.924844681834449 00E+04
  1.190084576082400 00E+06   1.683500282810187 00E+08  -1.722029255518489 00E+08
  1.167346336898621 00E+08  -1.143615096634352 00E+08
  7.461974759444938 00E+00   1.557485844869306 00E+01   4.152319275380631 00E+02
  3.078355551581359 00E+03   3.885832250136565 00E+03   4.855879319710295 00E+03
  6.733344117270694 00E+03   2.146969363173704 00E+04   1.343002889474730 00E+04
  5.336035995721526 00E+04   1.536157235243794 00E+05   3.547352339312884 00E+05
  5.752361517007196 00E+05   5.405075959672980 00E+05   7.458939451625072 00E+05
  1.682362043022550 00E+06   3.627282645180774 00E+06   3.630909927825949 00E+06
  4.480692176824208 00E+06   4.485172869001025 00E+06
 1.00000000
 0.00000000
C TL3 HAWK conductor
-1IN__2AA___1A                2.  0.00              -2
     12        4.090347064905766 0000E+02
  2.458905136002357 00E+03   1.006919164736511 00E+03   3.735522939590348 00E+03
  4.949108642839249 00E+03   1.809220174535260 00E+03   1.626389975440227 00E+03
  7.235902112013972 00E+02   9.980615716539589 00E+02   5.162740507080648 00E+02
  1.179555959812405 30E+04   6.126537567484807 00E+04   2.428839145698563 00E+06
  2.536132136180311 00E+00   3.678274032523600 00E+00   7.358252873864169 00E+00
  1.840262484773079 00E+01   3.580228449323820 00E+01   5.945599892702880 70E+01
  7.782171028000857 80E+01   1.297276129271710 00E+02   2.268054645514569 00E+02
  4.144249177996411 00E+03   2.155807371752076 00E+04   8.578379257482028 00E+05
     30        1.336438547538035 2000E-03
  9.689767135039627 00E-03   1.331202047681065 00E-02   2.185684272184940 00E-02
  2.196755642488093 00E-02   3.380780851909602 00E-02   4.334867710490493 00E-02
  6.761062469577644 00E-02   5.431404117894598 00E-02   1.414603897999451 00E-01
  9.110021576705052 00E-02   8.011809835464941 00E-02   1.124533345596398 00E-01
  1.380766929197304 00E-01   1.824636596236612 00E-01   2.640996184475385 00E+00
  3.878521965950510 40E+00   6.129388277338587 00E+00   1.909938103916744 00E+01
  7.965382431611341 80E+01   1.290566413684341 00E+02   9.016508406050309 00E+02
  7.999738273135195 00E+03   4.452554691419224 00E+03   1.422704347870997 00E+04
  1.508444031196744 00E+04   4.483753286444700 00E+05  -8.327008859757796 00E+07
  8.265957169738334 00E+07   6.594428351733515 00E+06  -6.475193678111386 00E+06
  3.726358376899904 00E+00   5.168941745399676 00E+00   8.238173615227803 00E+00
  8.843906554704869 00E+00   1.327556191420137 00E+01   1.687489131281257 00E+01
  2.326437780793673 00E+01   3.376754956259439 00E+01   2.831073743708522 00E+01
  3.461586842778176 00E+01   3.574188044772928 00E+01   4.534820477334193 00E+01
  5.772620259123803 00E+01   7.594041550273187 00E+01   1.048816581902940 00E+03
  1.644789901316057 00E+03   2.366463857329689 00E+03   3.560114877411563 00E+03
  5.839824342026182 00E+03   6.420728690925590 00E+03   1.138921741603524 00E+04
  5.355114569629604 00E+04   9.560936736291370 00E+04   7.296465027512175 00E+04
  1.317760606557312 00E+05   3.465693780607189 00E+05   6.032905696630392 00E+05
  6.038938602327026 00E+05   7.506352809546370 00E+05   7.513858733335579 00E+05
 1.00000000
 0.00000000
C TL4 HAWK conductor
-1A___1AA___2A                2.  0.00              -2
     12        4.090347064905766 0000E+02
  2.458905136002357 00E+03   1.006919164736511 00E+03   3.735522939590348 00E+03
  4.949108642839249 00E+03   1.809220174535260 00E+03   1.626389975440227 00E+03
  7.235902112013972 00E+02   9.980615716539589 00E+02   5.162740507080648 00E+02
  1.179555959812405 30E+04   6.126537567484807 00E+04   2.428839145698563 00E+06
  2.536132136180311 00E+00   3.678274032523600 00E+00   7.358252873864169 00E+00
  1.840262484773079 00E+01   3.580228449323820 00E+01   5.945599892702880 70E+01
  7.782171028000857 80E+01   1.297276129271710 00E+02   2.268054645514569 00E+02
  4.144249177996411 00E+03   2.155807371752076 00E+04   8.578379257482028 00E+05
     14        7.342684868065439 0000E-04
  3.297052335637034 00E-02   1.061101472002497 00E-01   1.496999969464306 00E-01
  4.540053146684312 00E-01   2.298947360516695 00E+01   2.002642254448675 00E+01
  3.985942374043684 00E+01   2.457595130087715 00E+03   3.071261016141588 00E+03
  3.625884304074284 00E+04   8.929668011873768 00E+04   8.324675197269716 00E+05
 -1.440685177348139 00E+08   1.431058822176750 00E+08
  5.878924993982997 00E+00   1.888865369771377 00E+01   2.733627912408763 00E+01
  8.329061708506586 00E+01   3.440444521443735 00E+03   4.074618319159675 00E+03
  3.669191945056970 00E+03   5.239[illegible]8782987440 00E+04   2.215984911169265 00E+04
  1.451879981525695 00E+05   2.730309631034551 00E+05   7.705431319403257 00E+05
  1.296899347555909 00E+06   1.298196246903463 00E+06
 1.00000000
 0.00000000
C TL5 HAWK conductor
-1X0044AA___1A                2.  0.00              -2
     12        4.090347064905766 0000E+02
  2.458905136002357 00E+03   1.006919164736511 00E+03   3.735522939590348 00E+03
  4.949108642839249 00E+03   1.809220174535260 00E+03   1.626389975440227 00E+03
  7.235902112013972 00E+02   9.980615716539589 00E+02   5.162740507080648 00E+02
  1.179555959812405 30E+04   6.126537567484807 00E+04   2.428839145698563 00E+06
  2.536132136180311 00E+00   3.678274032523600 00E+00   7.358252873864169 00E+00
  1.840262484773079 00E+01   3.580228449323820 00E+01   5.945599892702880 70E+01
```

```
  7.7821710280085780E+01   1.2972761292717100E+02   2.2680546455145690E+02
  4.1442491779864110E+03   2.1558073717520760E+04   8.5783792574820280E+05
     24         1.1674353929555290000E-04
 1.4033814424565100E-02   3.3970349003243010E-02   7.6905668181001290E+00
 3.5811010659989480E+01   4.5943485139443710E+01   2.8491735643264840E+02
 3.5338852110448310E+01  -2.0150784111073770E+02   6.3938248529242120E+01
 8.8853559813206180E+01   3.0102672982836120E+03   1.2606966598410270E+04
 6.1273815558461460E+04   2.3418990204234390E+05   1.3193599094607520E+05
 4.9467513445024570E+05   5.8456929976119680E+05   2.6874037494497080E+06
-2.5429380658398340E+05   2.0539686119399290E+06   2.6744229954245820E+07
-2.6425548822276120E+07   1.1374343825890450E+09  -1.1437627646857610E+09
 9.7070806337867260E+00   2.3577847439151700E+01   2.6582201854196230E+03
 1.2088084034093760E+04   1.5488346001106530E+04   1.9023092911426360E+04
 2.5317811533865220E+04   1.9029843866738980E+04   2.2448236168333970E+04
 3.1189251761645960E+04   1.2161489141559370E+05   2.5516622677975710E+05
 6.0862663209471530E+05   1.5077922919766400E+06   1.8379431968324100E+06
 2.8295021868272150E+06   4.4004391768181140E+06   7.8947247498317460E+06
 1.1155353840436825O0E+07   1.9376332875660550O0E+07   2.2513229020084280O0E+08
 2.2535742249104380O0E+07   3.2493229755722110E+07   3.2525722985477800E+07
C TL6 HAWK conductor
-1G___1AX0046A                2.  0.00              -2
     12         4.0903470649057660000E+02
 2.4589051360023570O0E+03   1.0069191647365110O0E+03   3.7355229395903480O0E+03
 4.9491086428392490O0E+03   1.8092201745352600O0E+03   1.6263899754402270O0E+03
 7.2359021120139720O0E+02   9.9806157165395890O0E+02   5.1627405070806480O0E+02
 1.1795559581240530O0E+04   6.1265375674848070O0E+04   2.4288391456985630O0E+06
 2.5361321361803110O0E+00   3.6782740325236000O0E+00   7.3582528738641690O0E+00
 1.8402624847730790O0E+01   3.5802284493238200E+01   5.9459989270288070O0E+01
 7.7821710280085780E+01   1.2972761292717100E+02   2.2680546455145690E+02
 4.1442491779864110E+03   2.1558073717520760E+04   8.5783792574820280E+05
     18         3.3358038754720500000E-05
 6.5887722409255740O0E+00   9.2113221746047390E+01   1.6423682191281080E+02
 3.0178407403080660O0E+02   4.0728514877113830O0E+02   7.2020971549401800E+02
 2.1160068828669650O0E+03   9.5441739397760270O0E+03  -7.2228283561231840O0E+05
 1.0059208008146480O0E+06   1.9061242592731810O0E+06   1.8683249264185610O0E+06
 1.2034035732374430O0E+07   3.9370274836321700O0E+07  -3.2956202374353530O0E+07
 1.1817207962102770O0E+08   9.0354875134119520O0E+10  -9.0495566761482650O0E+10
 2.3371251872899270O0E+03   3.2411038576230860O0E+04   5.8069296864557620O0E+04
 1.0495280561240230O0E+05   1.4968018477432050O0E+05   2.3961771850323940O0E+05
 3.3704846393224670O0E+05   4.2688804483029970O0E+05   2.1825853821176600O0E+06
 2.1590159257914460O0E+06   9.3528383821623330O0E+06   1.5716003127905200O0E+07
 4.2748384979229000O0E+07   6.3175353915514250O0E+07   6.8459854134571280O0E+07
 4.8514755209822910O0E+08   2.9449956354442660O0E+08   2.9479406310797130O0E+08
 1.00000000
 0.00000000
C TL7 HAWK conductor
-1A___2AX0042A                2.  0.00              -2
     12         4.0903470649057660000E+02
 2.4589051360023570O0E+03   1.0069191647365110O0E+03   3.7355229395903480O0E+03
 4.9491086428392490O0E+03   1.8092201745352600O0E+03   1.6263899754402270O0E+03
 7.2359021120139720O0E+02   9.9806157165395890O0E+02   5.1627405070806480O0E+02
 1.1795559581240530O0E+04   6.1265375674848070O0E+04   2.4288391456985630O0E+06
 2.5361321361803110O0E+00   3.6782740325236000O0E+00   7.3582528738641690O0E+00
 1.8402624847730790O0E+01   3.5802284493238200E+01   5.9459989270288070O0E+01
 7.7821710280085780E+01   1.2972761292717100E+02   2.2680546455145690E+02
 4.1442491779864110E+03   2.1558073717520760E+04   8.5783792574820280E+05
     24         1.1674353929555290000E-04
 1.4033814424565100E-02   3.3970349003243010E-02   7.6905668181001290E+00
 3.5811010659989480E+01   4.5943485139443710E+01   2.8491735643264840E+02
 3.5338852110448310E+01  -2.0150784111073770E+02   6.3938248529242120E+01
 8.8853559813206180E+01   3.0102672982836120E+03   1.2606966598410270E+04
 6.1273815558461460E+04   2.3418990204234390E+05   1.3193599094607520E+05
 4.9467513445024570E+05   5.8456929976119680E+05   2.6874037494497080E+06
-2.5429380658398340E+05   2.0539686119399290E+06   2.6744229954245820E+07
-2.6425548822276120E+07   1.1374343825890450E+09  -1.1437627646857610E+09
 9.7070806337867260E+00   2.3577847439151700E+01   2.6582201854196230E+03
 1.2088084034093760E+04   1.5488346001106530E+04   1.9023092911426360E+04
 2.5317811533865220E+04   1.9029843866738980E+04   2.2448236168333970E+04
 3.1189251761645960E+04   1.2161489141559370E+05   2.5516622677975710E+05
 6.0862663209471530E+05   1.5077922919766400E+06   1.8379431968324100E+06
 2.8295021868272150E+06   4.4004391768181140E+06   7.8947247498317460E+06
 1.1155353840436825O0E+07   1.9376332875660550O0E+07   2.2513229020084280O0E+08
 2.2535742249104380O0E+07   3.2493229755722110E+07   3.2525722985477800E+07
 1.00000000
 0.00000000
C TL8 HAWK conductor
-1X0049AG___2A                2.  0.00              -2
     12         4.0903470649057660000E+02
 2.4589051360023570O0E+03   1.0069191647365110O0E+03   3.7355229395903480O0E+03
 4.9491086428392490O0E+03   1.8092201745352600O0E+03   1.6263899754402270O0E+03
 7.2359021120139720O0E+02   9.9806157165395890O0E+02   5.1627405070806480O0E+02
 1.1795559581240530O0E+04   6.1265375674848070O0E+04   2.4288391456985630O0E+06
 2.5361321361803110O0E+00   3.6782740325236000O0E+00   7.3582528738641690O0E+00
 1.8402624847730790O0E+01   3.5802284493238200E+01   5.9459989270288070O0E+01
 7.7821710280085780E+01   1.2972761292717100E+02   2.2680546455145690E+02
 4.1442491779864110E+03   2.1558073717520760E+04   8.5783792574820280E+05
     12         5.0003139966653530000E-05
 2.2006505466685240O0E+02   1.9917133079851560O0E+01   3.0324587682859780O0E+02
 1.3331946931657280O0E+03   6.1065621476310030O0E+03   4.9073160329080080O0E+04
 8.2420464512539900O0E+04   1.7290284880488390O0E+06   1.9256172891326790O0E+06
 8.0633573471674240O0E+06   1.9562820318725090O0E+10  -1.9574677798459200O0E+10
 1.1365564306689630O0E+04   3.8106280777490570O0E+03   5.9144670933026060O0E+04
 2.2567118658118070O0E+05   2.8630129819420180O0E+05   9.0132352748558780O0E+05
 1.1637935076559950O0E+06   1.3601737115924910O0E+07   5.8380034486487250O0E+06
 2.6427968698049180O0E+07   1.1148336231140280O0E+08   1.1159484567371430O0E+08
 1.00000000
 0.00000000
C TL9 HAWK conductor
-1X0046AX0044A                2.  0.00              -2
     12         4.0903470649057660000E+02
 2.4589051360023570O0E+03   1.0069191647365110O0E+03   3.7355229395903480O0E+03
 4.9491086428392490O0E+03   1.8092201745352600O0E+03   1.6263899754402270O0E+03
 7.2359021120139720O0E+02   9.9806157165395890O0E+02   5.1627405070806480O0E+02
 1.1795559581240530O0E+04   6.1265375674848070O0E+04   2.4288391456985630O0E+06
 2.5361321361803110O0E+00   3.6782740325236000O0E+00   7.3582528738641690O0E+00
 1.8402624847730790O0E+01   3.5802284493238200E+01   5.9459989270288070O0E+01
 7.7821710280085780E+01   1.2972761292717100E+02   2.2680546455145690E+02
 4.1442491779864110E+03   2.1558073717520760E+04   8.5783792574820280E+05
     19         2.1677912323901730000E-04
 2.5668812019169910O0E-02   7.4573586542535580O0E-02   2.6158242533535060E+00
 1.5821400034897010O0E+01   2.0157791621581910O0E+01   2.3095225558142560E+01
 3.7513625899986710O0E+01   9.5468282926668350O0E+01   6.0412567556937340O0E+02
 9.7648615523174430O0E+03   1.4706488559565730O0E+04   3.5425302215796070O0E+05
 3.5923220693707160O0E+04   2.7768142142824660O0E+05   7.8167462058354530O0E+05
 7.7471224487183280O0E+09  -7.5980069769264870O0E+09   1.3551323622496550O0E+10
-1.3701913896819820O0E+10
 9.2789686020372990O0E+00   2.7107207914849560E+01   9.5104437069529050O0E+02
 5.5849222080702620O0E+03   7.2943745131051480O0E+03   8.3167404305381530O0E+03
 1.3336790252125830O0E+04   3.6099635343242400E+04   2.6834670922832760O0E+04
 1.5436613667813580O0E+05   1.9357236334150650O0E+05   8.8420482900030070O0E+05
 1.0837417027655030O0E+06   1.7693252262584120O0E+06   2.7931417931203100E+06
 1.2403798307821410O0E+07   1.2416202106129210O0E+07   1.0787161250841890O0E+07
 1.0797948412092740O0E+07
 1.00000000
 0.00000000
C TL10 HAWK conductor
-1X0046AA___1A                2.  0.00              -2
     12         4.0903470649057660000E+02
 2.4589051360023570O0E+03   1.0069191647365110O0E+03   3.7355229395903480O0E+03
 4.9491086428392490O0E+03   1.8092201745352600O0E+03   1.6263899754402270O0E+03
 7.2359021120139720O0E+02   9.9806157165395890O0E+02   5.1627405070806480O0E+02
 1.1795559581240530O0E+04   6.1265375674848070O0E+04   2.4288391456985630O0E+06
 2.5361321361803110O0E+00   3.6782740325236000O0E+00   7.3582528738641690O0E+00
 1.8402624847730790O0E+01   3.5802284493238200E+01   5.9459989270288070O0E+01
 7.7821710280085780E+01   1.2972761292717100E+02   2.2680546455145690E+02
 4.1442491779864110E+03   2.1558073717520760E+04   8.5783792574820280E+05
     19         4.4385143983056110000E-04
 1.3846733280850070O0E-02   6.3612025723329370E-02   4.2065566089219500E-02
 8.4294061998856490O0E-02   2.7830324593105840O0E-01   8.6339420458983720O0E+00
 1.0416845703890280O0E+01   2.2640770995110300O0E+01   1.6322421017560130O0E+02
 5.1764139806445930O0E+03   4.7461012003364690O0E+03   2.6632381229083540O0E+04
 1.2830609309403920O0E+05   4.4148861677650630O0E+04   1.1532636208966570O0E+06
-8.2870306359694590O0E+07   8.1501201114640980O0E+07   9.0724858988122100E+03
-2.4461108141898670O0E+03
 5.0543630218745620O0E+00   1.9228451428876380O0E+01   1.9481517270651050O0E+01
 3.1383007112171700O0E+01   1.0383346454392840O0E+02   3.0841295424481410O0E+03
```

```
   3.88989447310648000E+03   8.89831389746010600E+03   7.36254554745701100E+03
   1.10802992274832600E+05   5.06775394330296000E+04   4.12837881301153100E+05
   3.30469476516399400E+05   5.71943555824721700E+05   1.39415645997757700E+06
   2.64047008033511200E+06   2.64311055041544700E+06   3.73512434447443600E+06
   3.73885946881890500E+06
  1.00000000
  0.00000000
C TL11 HAWK conductor
-1X0042AX0049A                2.  0.00              -2
     12      4.09034706490576600000E+02
   2.45890513600235700E+03   1.00691916473651100E+03   3.73552293959034800E+03
   4.94910864283924900E+03   1.80922017453526000E+03   1.62638997544022700E+03
   7.23590211201397200E+02   9.98061571653958900E+02   5.16274050708064800E+02
   1.17955595981240530OE+04   6.12653756748480700E+04   2.42883914569856300E+06
   2.53613213618031100E+00   3.67827403252360000E+00   7.35825287386416900E+00
   1.84026248477307900E+01   3.58022844932382000E+01   5.94559892702880700E+01
   7.78217102800857800E+01   1.29727612927171000E+02   2.26805464551456900E+02
   4.14424917798641100E+03   2.15580737175207600E+04   8.57837925748202800E+05
     21      1.40129890129590700000E-04
   2.82409465040976800E-02   5.73666755142106500E-01   2.17536291440966400E+01
   3.46969984772486200E+01   2.86322638506245000E+01   2.72358421123721800E+01
   3.30384774679833600E+01   5.44278581191543800E+01   -7.83622740098267300E+02
   9.53611012878148500E+03   4.52847893308504100E+04   2.09702191470714500E+05
   7.34012430467903700E+05   9.48685976801616600E+04   1.09528829765371700E+06
   1.27602110708312400E+06   4.46680320404957500E+05   8.92491202129325800E+06
  -8.81159104863197700E+06   1.10155759833908800E+09   -1.10558172992019600E+09
   1.05642363611503200E+01   2.15117728119521300E+02   8.00177968562786300E+03
   1.17453901961811200E+04   1.25836048886094200E+04   9.82315069880778500E+03
   1.22450233355681700E+04   2.04493234107368700E+04   1.35561763828908700E+05
   1.31015066778640600E+05   4.65519105469535800E+05   1.07897979236858500E+06
   2.39840255363253400E+06   2.93078299179311300E+06   4.87134948866919700E+06
   8.30249729851572600E+06   1.23018882615654100E+07   2.64849913464487000E+08
   2.65114763377951000E+08   2.15065384249761900E+07   2.15280449634011800E+07
  1.00000000
  0.00000000
C TL12 HAWK conductor
-1A___1AX0049A                2.  0.00              -2
     12      4.09034706490576600000E+02
   2.45890513600235700E+03   1.00691916473651100E+03   3.73552293959034800E+03
   4.94910864283924900E+03   1.80922017453526000E+03   1.62638997544022700E+03
   7.23590211201397200E+02   9.98061571653958900E+02   5.16274050708064800E+02
   1.17955595981240530OE+04   6.12653756748480700E+04   2.42883914569856300E+06
   2.53613213618031100E+00   3.67827403252360000E+00   7.35825287386416900E+00
   1.84026248477307900E+01   3.58022844932382000E+01   5.94559892702880700E+01
   7.78217102800857800E+01   1.29727612927171000E+02   2.26805464551456900E+02
   4.14424917798641100E+03   2.15580737175207600E+04   8.57837925748202800E+05
     30      1.25283427372755600000E-03
   9.08757311529024800E-03   1.24921965405884400E-02   2.04684013014478000E-02
   2.06615327397690500E-02   3.31759823948442500E-02   4.20176524185137500E-02
   4.23787906612785400E-02   1.01879007311903300E-01   7.78394046512750100E-02
  -2.02349559986073500E-01   3.10375115852991000E-01   7.86215129679403700E-02
   1.05978609404068000E-01   1.27267670612558800E-01   1.71998694905671200E-01
   2.79964668391564400E+00   3.63380499208956800E+00   5.74442869220301300E+00
   1.78597780990942000E+01   7.03758038651224200E+01   1.33287316651978400E+02
   8.08711023086552800E+02   7.71126319206399800E+03   1.70597935435073100E+04
   1.55185819572756500E+04   3.50512237615357300E+05   9.59090873569393800E+11
  -5.01489359681364200E+12   5.01280173512390700E+12   -9.56999403724883900E+11
   3.72666260258944100E+00   5.16936374583739600E+00   8.23884462804631900E+00
   8.84463026449296800E+00   1.37960479431099500E+01   1.82225815658297100E+01
   1.71156600972840500E+01   2.87974024832269900E+01   3.34578016143075200E+01
   2.91435191536744300E+01   2.91280076907280200E+01   3.57447959438362200E+01
   4.53519895960392500E+01   5.66336588690298900E+01   7.59465867562610400E+01
   1.17688664181622500E+03   1.64491287601993100E+03   2.36665475861263800E+03
   3.56068851790910000E+03   5.84176701978976700E+03   6.67630768468816300E+03
   1.09890912038616700E+04   8.27565673224283100E+04   6.29060985548070500E+04
   1.32473146492676300E+05   3.40564995673745100E+05   7.37142997742820400E+05
   7.37880140740563900E+05   7.38228667513775000E+05   7.38966896181291800E+05
  1.00000000
  0.00000000
/SWITCH
C < n 1>< n 2>< Tclose ><Top/Tde ><   Ie    ><Vf/CLOP ><  type  >
  X0017ASZ__2A        .05       .15                                          0
  SZ__1ASZTMPA                                  MEASURING                    1
```

```
/SOURCE
C < n 1><>< Ampl.  >< Freq.  ><Phase/T0><   A1   ><   T1   >< TSTART >< TSTOP  >
14X0007A 0       195.        60.       10.5                          -1.        1.
14X0010A 0       200.        60.       20.2                          -1.        1.
14X0030A 0       195.        60.       -17.                          -1.        1.
14X0035A 0       200.        60.       -6.8                          -1.        1.
/OUTPUT
  SZ__1ASZ__2A
BLANK BRANCH
BLANK SWITCH
BLANK SOURCE
BLANK OUTPUT
BLANK PLOT
BEGIN NEW DATA CASE
BLANK
```

B.4 ATP Data File for Robust TLNE Model

B.4.1 Capacitor $C1$ Switching Case

```
BEGIN NEW DATA CASE
C ------------------------------------------------------------
C Xin Nie
C Power Engineering Group
C Dept. of Electrical and Computer Engineering
C University of Alberta
C September 22, 2004
C ------------------------------------------------------------
POWER FREQUENCY                    60.
$DUMMY, XYZ000
C  dT  >< Tmax >< Xopt >< Copt >
   2.E-5      .15
     50       1       0       0       0       0       0       0       1       0
C        1       2       3       4       5       6       7       8
C 345678901234567890123456789012345678901234567890123456789012345678901234567890
/BRANCH
C < n 1>< n 2><ref1><ref2>< R  >< L  >< C  >
C < n 1>< n 2><ref1><ref2>< R  >< A  >< B  ><Leng><><>0
C ********************************* Alpha Mode *********************************
C *************
C study zone
C *************
  SZ__1ASZTMPA             1.E-9                                              1
  SZ__1A                    200.  130.                                        0
  SZ__2A                    650.  250.                                        0
  IN__1A                   1200.  500.                                        0
  IN__2A                   2150.  380.                                        0
  X0018A                              8.                                     0
C HAWK conductor
-1SZTMPAIN__1A                2.  0.00              -2 1
     12      4.09034706490576600000E+02
   2.45890513600235700E+03   1.00691916473651100E+03   3.73552293959034800E+03
   4.94910864283924900E+03   1.80922017453526000E+03   1.62638997544022700E+03
   7.23590211201397200E+02   9.98061571653958900E+02   5.16274050708064800E+02
   1.17955595981240530OE+04   6.12653756748480700E+04   2.42883914569856300E+06
   2.53613213618031100E+00   3.67827403252360000E+00   7.35825287386416900E+00
   1.84026248477307900E+01   3.58022844932382000E+01   5.94559892702880700E+01
   7.78217102800857800E+01   1.29727612927171000E+02   2.26805464551456900E+02
   4.14424917798641100E+03   2.15580737175207600E+04   8.57837925748202800E+05
     19      4.00519444461175000000E-04
   1.56976974488308500E-02   5.96931801697355600E-02   7.65710632981902200E-02
   1.23025310978241300E-01   7.74362618176230500E+00   1.15631326792783200E+01
   1.51793395253545000E+01   2.45460099809200000E+01   2.17562524566291500E+02
   4.75157557302470000E+03   7.17277231675696400E+03   2.15420855113000800E+02
   1.92780440925090000E+05   1.00130204456972800E+05   6.67694141945910300E+05
   4.49850806862179600E+09  -4.59220311195214400E+09   3.24980116228040600E+09
```

```
  -3.157079140375575O0E+09
   5.460067998651540O0E+00   2.063829992047784O0E+01   2.694103636262369O0E+01
   4.366175083258754O0E+01   2.728891534109220O0E+03   3.969698516654513O0E+03
   5.156233595789232O0E+03   1.038658273690606O0E+04   9.267324595579616O0E+03
   1.124694586165610O0E+05   6.932342002009935O0E+04   4.989208395223323O0E+05
   4.157544530856369O0E+05   7.764223805858483O0E+05   1.439448374462287O0E+06
   4.025219577684497O0E+06   4.029244797262184O0E+06   4.381295402519039O0E+06
   4.385676697921551O0E+06
  1.00000000
  0.00000000
C DRAKE conductor
-1SZTMPASZ__2A                2.  0.00              -2 1
      11       3.938973277035573O000E+02
   2.271090787356759O0E+03   2.468738369802207O0E+03   2.767283308844692O0E+03
   9.272013278576356O0E+02   8.136523854897157O0E+02   4.843864533476302O0E+02
   6.006604547470770O0E+02   3.191230068771898O0E+02   1.474134983976601O0E+04
   5.357097306002596O0E+04   2.761989693560646O0E+06
   2.565204294186215O0E+00   6.238679152406237O0E+00   1.373155019487035O0E+01
   2.444115662193466O0E+01   3.882345993654990O0E+01   5.220646860706471O0E+01
   8.949853843076818O0E+01   1.612485445474848O0E+02   6.100242815499042O0E+03
   2.242497251791498O0E+04   1.159710946925651O0E+06
     19       4.539154979287958O000E-04
   1.712759170053953O0E-02   3.986913373903335O0E-02   1.608642600299830E-01
   6.890036182906788O0E+00   8.786329257170928O0E+00   1.187084000840551O0E+01
   1.466504281283071O0E+01   4.551321690537482O0E+01   1.926819811653192O0E+02
   2.213264955675185O0E+03   1.110004761321304O0E+04   9.214328328701855O0E+04
   2.817765442301622O0E+05   1.678419416532637O0E+04   1.225350369761673O0E+06
  -3.168762404556547O0E+07   2.934646328211559O0E+07   3.778887191634843O0E+07
  -3.689307291564467O0E+07
   6.299201827904183O0E+00   1.475354424616955O0E+01   5.988920629316388O0E+01
   2.468781047206587O0E+03   3.246882256715642O0E+03   4.205475680336974O0E+03
   5.301662381272396O0E+03   1.665669470442428O0E+04   8.865719886677556O0E+03
   4.657461284769958O0E+04   1.133980748245041O0E+05   4.435156084732816O0E+05
   3.984178416006317O0E+05   5.620686799638157O0E+05   1.446828729109786O0E+06
   2.871519673200630O0E+06   2.874391192873833O0E+06   3.654271889652570O0E+06
   3.657926161542183O0E+06
  1.00000000
  0.00000000
C HAWK conductor
-1SZ__2AIN__2A                2.  0.00              -2 1
      12       4.090347064905766O000E+02
   2.458905136002357O0E+03   1.006919164736511O0E+03   3.735522939590348O0E+03
   4.949108642839249O0E+03   1.809220174535260O0E+03   1.626389975440227O0E+03
   7.235902112013972O0E+02   9.980615716539580O0E+02   5.162740507080648O0E+02
   1.179555958124053O0E+04   6.126537567484807O0E+04   2.428839145698563O0E+06
   2.536132136180311O0E+00   3.678274032523600O0E+00   7.358252873864169O0E+00
   1.840262484773079O0E+01   3.580228449323820O0E+01   5.945998927028807O0E+01
   7.782171028008578O0E+01   1.297276129271710O0E+02   2.268054645514569O0E+02
   4.144249177986411O0E+03   2.155807371752076O0E+04   8.578379257482028O0E+05
     19       5.505543845357618O000E-04
   9.794656058558596O0E-03   4.062127331237850O0E-02   7.448549031790969O0E-02
   4.002298701448773O0E-02   6.562545693306770O0E-02   7.793456900274408O0E-02
   7.111514662870558O0E-01   6.740052368561383O0E+00   8.016207673581198O0E+00
   1.172206700749057O0E+01   3.502971302755199O0E+01   7.054946753887513O0E+01
   7.136004868170412O0E+02   4.435453461485819O0E+03   6.117812783015694O0E+04
   6.793005712445018O0E+04   7.497647220639265O0E+05   2.364694593071054O0E+07
  -2.453110096882049O0E+07
   3.945877150681552O0E+00   1.570359534061049O0E+01   3.033936660197176O0E+01
   1.670661222945046O0E+01   2.608700546592988O0E+01   3.324921072848005O0E+01
   2.919886416106979O0E+02   2.664618156511500O0E+03   3.319650639220727O0E+03
   4.566096983044453O0E+03   7.473356722831026O0E+03   6.699036297016883O0E+03
   1.759025397917742O0E+04   5.336861955214478O0E+04   2.154763142183993O0E+05
   3.138247765040839O0E+05   9.375308613961636O0E+05   1.823251480344845O0E+06
   1.825074731825187O0E+06
  1.00000000
  0.00000000
C ************
C surface layer
C ************
C DRAKE conductor
-1IN__1AA____1                2.  0.00              -2 1
       8       3.750238347789160700E+002
   5.118235030896182 9E+005  -7.722040015021633 0E+004   5.383608138738141 9E+005
  -1.313078711186444 9E+006   1.410685475873848 0E+006  -8.705674434205368 2E+005
   3.973888619307641 6E+005  -8.108690203008463 1E+004
   1.094578254690202 1E+004   8.949853843076817 8E+001   5.220646860706471 2E+001
   3.665170960815073 2E+001   2.539735097738093 8E+001   1.373155019487035 0E+001
   6.003796790214838 8E+000   2.665560995982649 0E+000
     10       5.009725535275262 500E-004
   1.627086390646989 9E+008  -1.643159600115771 9E+008   1.421420445547349 1E+006
   1.088526143470841 1E+005   6.417452860057933 1E+004   1.156280883709591 9E+004
   1.049866538312715 0E+003   1.230140945629611 1E+002   1.374426022997039 1E+002
   2.263079711700468 1E-001
   2.185199651861968 9E+006   2.183016635226741 0E+006   1.279728252208564 9E+006
   4.181491189026482 8E+005   2.579423631465169 0E+005   1.128834522077870 9E+005
   2.536520091355888 1E+004   5.894698410030385 3E+003   1.300538441909239 0E+004
   2.222561607240679 0E+001
  1.00000000
  0.00000000
C DRAKE conductor
-1IN__2AA____2                2.  0.00              -2 1
       8       3.939032973350055600E+002
   1.051104549293156 9E+005   2.416454714841457 9E+002  -1.501718695581012 3E+004
   6.347604649385272 9E+004  -8.013588947233212 8E+004   4.997037868767719 2E+004
  -8.553726482229867 5E+003   3.388080857312269 2E+002
   1.094578254690202 1E+004   8.949853843076817 8E+001   5.220646860706471 2E+001
   3.665170960815073 2E+001   2.539735097738093 8E+001   1.373155019487035 0E+001
   6.003796790214838 8E+000   2.665560995982649 0E+000.
     12       4.004260509612045000E-004
   2.982141845328816 2E+008  -3.004419367481235 9E+008   1.949537357407647 0E+006
   1.811905112397948 1E+005   4.528709501300411 6E+004   4.941671595339536 7E+004
   1.852339603200568 0E+003   3.633340265252887 8E+002   5.801399406487576 7E+001
   3.926894860183546 1E+001   7.517152252934754 3E+000   6.190596751150582 3E+002
   3.192360503868841 1E+006   3.189171332536302 0E+006   1.851766094690880 0E+006
   6.084430081879762 0E+005   3.082953280765123 2E+005   2.481812708164527 1E+005
   4.078984771343108 9E+004   1.596161925595068 0E+004   9.920004276656654 8E+003
   6.796981390611043 0E+003   1.343825816901834 1E+003   1.110096612523556 9E+001
  1.00000000
  0.00000000
C HAWK conductor
-1IN__2AA____1                2.  0.00              -2 1
      12       4.090334894525543100E+002
   2.428846130335301 6E+006   6.127366368720508 0E+004   1.179190694404237 6E+004
   5.195656970932087 7E+002   3.323641859038797 9E+002   5.716435736749897 8E+003
   3.263400416932691 2E+004  -1.558652431632784 4E+005   4.841214768044872 3E+005
  -1.238474876016972 0E+006   1.394418985543663 1E+006  -5.055791666107732 5E+005
   8.578379257482028 3E+005   2.155807371752076 1E+004   4.144249177986411 3E+003
   2.268054645514569 1E+002   1.297276129271710 1E+002   7.782171028008578 2E+001
   5.945998927028806 9E+001   3.580228449323819 7E+001   1.840262484773079 2E+001
   7.358252873864168 8E+000   3.678274032523599 6E+000   2.536132136180310 9E+000
     11       1.335538585935080000E-003
  -1.774834586950440 7E+009   1.750773288229799 6E+009   2.402386640005841 1E+007
   2.178189364354504 1E+004   1.451000819099311 0E+004   5.993323234369857 8E+002
   5.390570562699196 0E+002   9.029102553129663 1E-001   2.888288587005956 0E-001
   1.824108926341173 9E-001   6.363082288528597 3E-002
   4.945597143135687 2E+005   4.940656486664945 8E+005   5.285955714516588 5E+005
   9.379049601890085 6E+004   5.828185234595023 9E+004   7.049536284446858 2E+003
   1.409025636209587 0E+004   9.202430116657631 9E+001   2.926633735285199 1E+001
   1.782884187530103 0E+001   6.180901318245266 1E+000
  1.00000000
  0.00000000
C ********************************** Beta Mode ********************************
C ************
C study zone
C ************
  SZ__1BSZTMPB              1.E-9                                              1
  SZ__1B                     200.  130.                                        0
  SZ__2B                     650.  250.                                        0
  IN__1B                    1200.  500.                                        0
  IN__2B                    2150.  380.                                        0
  X0018B                             8.
C HAWK conductor
-1SZTMPBIN__1B                2.  0.00              -2 1
      12       4.090347064905766O000E+02
   2.458905136002357O0E+03   1.006919164736511O0E+03   3.735522939590348O0E+03
   4.949108642839249O0E+03   1.809220174535260O0E+03   1.626389975440227O0E+03
```

```
  7.2359021120139720OE+02   9.9806157165395890OE+02   5.1627405070806480OE+02
  1.1795595981240530OE+04   6.1265375674848070OE+04   2.4288391456985630OE+06
  2.5361321361803110OE+00   3.6782740325236000OE+00   7.3582528738641690OE+00
  1.8402624847730790OE+01   3.5802284493238200OE+01   5.9459989270288070OE+01
  7.7821710280085780OE+01   1.2972761292717100OE+02   2.2680546455145690OE+02
  4.1442491779864110OE+03   2.1558073717520760OE+04   8.5783792574820280OE+05
     19        4.0051944446117500000E-04
  1.5697697448830850OE-02   5.9693180169735560OE-02   7.6571063298190220OE-02
  1.2302531097824130OE-01   7.7436261817623050OE+00   1.1563132679278320OE+01
  1.5179339525354500OE+01   2.4546009980920000OE+01   2.1756252456629150OE+02
  4.7515755730247000OE+03   7.1727723167569640OE+03   2.1542085511300080OE+02
  1.9278044092509000OE+05   1.0013020445697280OE+05   6.6769414194591030OE+05
  4.4985080686217960OE+09  -4.5922031119521440OE+09   3.2498011622804060OE+09
 -3.1570791403755750OE+09
  5.4600679986515400OE+00   2.0638299920477840OE+01   2.6941036362623690OE+01
  4.3661750832587540OE+01   2.7288915341019220OE+03   3.9696985166545130OE+03
  5.1562335957892320OE+03   1.0386582736906060OE+04   9.2673245955796160OE+03
  1.1246945861656100OE+05   6.9323402002009350OE+04   4.9892083952232300OE+05
  4.1575445308556360OE+05   7.7642238058548340OE+05   1.4394483744622870OE+06
  4.0252159577684490OE+06   4.0292447972621840OE+06   4.3812954025190390OE+06
  4.3856766979215510OE+06
 1.00000000
 0.00000000
C DRAKE conductor
-1SZTMPBSZ__2B                2.  0.00              -2 1
     11        3.9389732770355730000E+02
  2.2710907873567590OE+03   2.4687383698022070OE+03   2.7672833088446920OE+03
  9.2720132785763560OE+02   8.1365238548971570OE+02   4.8438645334763020OE+02
  6.0066045474707700OE+02   3.1912300687718980OE+02   1.4741349839766010OE+04
  5.3570973060025960OE+04   2.7619896935606460OE+06
  2.5652042941862150OE+00   6.2386791524062370OE+00   1.3731550194870350OE+01
  2.4441156621934660OE+01   3.8823459936549900OE+01   5.2204648607064710OE+01
  8.9498538430768180OE+01   1.6124854454748480OE+02   6.1002428154998420OE+03
  2.2424972517914980OE+04   1.1597109469265610OE+06
     19        4.5391549792879580000E-04
  1.7127591700539550OE-02   3.9869133739033350OE-02   1.6086426002998300OE-01
  6.8900361829067880OE+00   8.7863292571709280OE+00   1.1870840008405510OE+01
  1.4665042812830710OE+01   4.5513216905374820OE+01   1.9268198116531920OE+02
  2.2132649556751850OE+03   1.1100047613213040OE+04  -9.2143283287018550OE+04
  2.8177654423016220OE+05   1.6784194165326370OE+04   1.2253503697616730OE+06
 -3.1687624045565470OE+07   2.9346463282115590OE+07   3.7788871916348430OE+07
 -3.6893072915644670OE+07
  6.2992018279041830OE+00   1.4753544246169550OE+01   5.9889206293163880OE+01
  2.4687810472065870OE+03   3.2468822567156420OE+03   4.2054756803369740OE+03
  5.3016623812723960OE+03   1.6656694704424280OE+04   8.8657198866775560OE+03
  4.6574612847695800OE+04   1.1339807482504100OE+05   4.4351560847328160OE+05
  3.9841784160063170OE+05   5.6206867996381570OE+05   1.4468287291097860OE+06
  2.8715196732006300OE+06   2.8743911928738300OE+06   3.6542718896525270OE+06
  3.6579261615421830OE+06
 1.00000000
 0.00000000
C HAWK conductor
-1SZ__2BIN__2B                2.  0.00              -2 1
     12        4.0903470649057660000E+02
  2.4589051360023570OE+03   1.0069191647365110OE+03   3.7355229395903480OE+03
  4.9491086428392490OE+03   1.8092201745352600OE+03   1.6263899754402270OE+03
  7.2359021120139720OE+02   9.9806157165395890OE+02   5.1627405070806480OE+02
  1.1795595981240530OE+04   6.1265375674848070OE+04   2.4288391456985630OE+06
  2.5361321361803110OE+00   3.6782740325236000OE+00   7.3582528738641690OE+00
  1.8402624847730790OE+01   3.5802284493238200OE+01   5.9459989270288070OE+01
  7.7821710280085780OE+01   1.2972761292717100OE+02   2.2680546455145690OE+02
  4.1442491779864110OE+03   2.1558073717520760OE+04   8.5783792574820280OE+05
     19        5.5055438453576180000E-04
  9.7946560585585960OE-03   4.0621273312378500OE-02   7.4485490317909690OE-02
  4.0022987014487730OE-02   6.5625456493340670OE-02   7.7934569002744080OE-02
  7.1115146628705580OE-01   6.7400523685613830OE+00   8.0162076735811980OE+00
  1.1722067007490570OE+01   3.5029713027551990OE+01   7.0549467538875130OE+01
  7.1360048681704120OE+02   4.4354534614858190OE+03   6.1178127830156940OE+04
  6.7930057124450180OE+04   7.4976472206392650OE+05   2.3646945930710540OE+07
 -2.4531100968820490OE+07
  3.9458771506815520OE+00   1.5703595340610490OE+01   3.0339366601971760OE+01
  1.6706612229450460OE+01   2.6087005465929880OE+01   3.3249210728480050OE+01
  2.9198864161069790OE+02   2.6646181565115000OE+03   3.3196506392207270OE+03
  4.5660969830444530OE+03   7.4733567228310260OE+03   6.6990362970168830OE+03
  1.7590253979177420OE+04   5.3368619552144780OE+04   2.1547631421839930OE+05
  3.1382477605408390OE+05   9.3753086139616360OE+05   1.8232514803448450OE+06
  1.8250747318251870OE+06
 1.00000000
 0.00000000
C *************
C surface layer
C *************
C DRAKE conductor
-1IN__1BB____1                2.  0.00              -2 1
     8        3.7502383477891600700E+002
  5.1182350308961829E+005  -7.7220400150216330E+004   5.3836081387381419E+005
 -1.3130787111864449E+006   1.4106854758738480E+006  -8.7056744342053682E+005
  3.9738886193076416E+005  -8.1086902030084631E+004
  1.0945782546902021E+004   8.9498538430768178E+001   5.2204648607064712E+001
  3.6651709608150732E+001   2.5397350977380938E+001   1.3731550194870350E+001
  6.0037967902148388E+000   2.6655609959826490E+000
     10        5.0097255352752625000E-004
  1.6270863906469899E+008  -1.6431596001157719E+008   1.4214204455473491E+006
  1.0885261434708411E+005   6.4174528600579331E+004   1.1562808837095919E+004
  1.0498665383127150E+003   1.2301409456296111E+002   1.3744260229970391E+002
  2.2630797117004681E-001
  2.1851996518619689E+006   2.1830166352267410E+006   1.2797282522085649E+006
  4.1814911890264828E+005   2.5794236314651690E+005   1.1288345220778709E+005
  2.5365200913558881E+004   5.8946984100303853E+003   1.3005384419092390E+004
  2.2225616072406790E+001
 1.00000000
 0.00000000
C DRAKE conductor
-1IN__2BB____2                2.  0.00              -2 1
     8        3.9390329733500556000E+002
  1.0511045492931569E+005   2.4164547148414579E+002  -1.5017186955810123E+004
  6.3476046493852729E+004  -8.0135889472332128E+004   4.9970378687677192E+004
 -8.5537264822298675E+003   3.3880808573122692E+002
  1.0945782546902021E+004   8.9498538430768178E+001   5.2204648607064712E+001
  3.6651709608150732E+001   2.5397350977380938E+001   1.3731550194870350E+001
  6.0037967902148388E+000   2.6655609959826490E+000
     12        4.0042605096120450000E-004
  2.9821418453288162E+008  -3.0044193674812359E+008   1.9495373574076470E+006
  1.8119051123979481E+005   4.5287095013004116E+004   4.9416715953395367E+004
  1.8523396032005680E+003   3.6333402652528878E+002   5.8013994064875767E+001
  3.9268948601835461E+001   7.5171522529347543E+000   6.1905967511505823E-002
  3.1923605038688411E+006   3.1891713325363020E+006   1.8517660946908800E+006
  6.0844300818797620E+005   3.0829532807651232E+005   2.4818127081645271E+005
  4.0789847713431889E+004   1.5961619255950680E+004   9.9200424766566548E+003
  6.7969813906110430E+003   1.3438258169018341E+003   1.1100966125235569E+001
 1.00000000
 0.00000000
C HAWK conductor
-1IN__2BB____1                2.  0.00              -2 1
     12        4.0903348945255431000E+002
  2.4288461303353016E+006   6.1273663687205080E+004   1.1791906944042370E+004
  5.1956569709320877E+002   3.3236418590387979E+002   5.7164357367498978E+003
  3.2634004169326912E+004  -1.5586524316327844E+005   4.8412147680448723E+005
 -1.2384748760169270E+006   1.3944189855436631E+006  -5.0557916661077325E+005
  8.5783792574820283E+005   2.1558073717520761E+004   4.1442491779864113E+003
  2.2680546455145691E+002   1.2972761292717101E+002   7.7821710280085782E+001
  5.9459989270288069E+001   2.5802284493238197E+001   1.8402624847730792E+001
  7.3582528738641688E+000   3.6782740325235999E+000   2.5361321361803109E+000
     11        1.3355385859350800000E-003
 -1.7748345869520440E+009   1.7507732888229799E+009   2.4023866400058411E+007
  2.1781893643545041E+004   1.4510008190993110E+004   5.9933233243698578E+002
  5.3905705626991960E+002   9.0291025531296631E-001   2.8882885870059560E-001
  1.8241089263411739E+001   6.3630822885285973E-002
  4.9455971431356872E+005   4.9406564866490458E+005   5.2859557145165885E+005
  9.3790496018900856E+004   5.8281852345950239E+004   7.0495362844468582E+003
  1.4090256362095870E+004   9.2024301665763190E+001   2.9266337352851991E+001
  1.7828841875301030E+001   6.1809013182452661E+000
 1.00000000
 0.00000000
C ********************************* Alpha Mode *********************************
C *************
```

```
C deep region
C ************
C BEGIN FDNE
$VINTAGE,1
C <BUS1><BUS2><BUS3><BUS4><   OHM        ><   milliH     ><   microF     >
C
C (1,1)
  A____1                    1.179990e+002
  A____1                   -1.817485e+001  -1.663768e+003
  A____1                    1.139488e+001   2.460232e+002
  A____1                    1.235152e+001   1.618993e+002
  A____1                    1.198652e+003   5.511484e+002
  A____1                   -1.989105e+002  -1.174363e+001
  A____1A 6__1              6.337633e+000   6.022723e+001
  A 6__1                    4.738628e+002
  A 6__1                                                  1.181832e+001
  A____1A 8__1              1.102904e+001   2.229382e+002
  A 8__1                    2.072046e+004
  A 8__1                                                  8.153085e-001
  A____1A A__1              1.013795e+001   7.765019e+001
  A A__1                    1.249981e+006
  A A__1                                                  7.261159e-001
  A____1A C__1              2.569920e+002   3.941270e+002
  A C__1                   -1.787782e+004
  A C__1                                                  1.139597e-001
  A____1A E__1             -1.620042e+002   2.269678e+002
  A E__1                    1.168918e+004
  A E__1                                                  8.943689e-002
  A____1A10__1              2.320907e+004   1.290952e+003
  A10__1                   -2.935271e+004
  A10__1                                                  1.945724e-003
  A____1A12__1              4.713920e+001   6.899455e+001
  A12__1                   -2.446764e+004
  A12__1                                                  8.993800e-002
  A____1A14__1             -2.288147e+006  -1.622627e+004
  A14__1                    2.321225e+006
  A14__1                                                 -3.060986e-006
C (1,2)
  A____1A____2             -6.862856e+003
  A____1A____2              5.790700e+001   5.300940e+003
  A____1A____2             -3.452911e+001  -7.455070e+002
  A____1A____2              1.627967e+001   2.133881e+002
  A____1A____2              3.760290e+003   1.729007e+003
  A____1A____2              2.107372e+004   1.244187e+003
  A____1A 6_12              3.509147e+004  -5.910586e+004
  A 6_12A____2             -1.309887e+005
  A 6_12A____2                                           -8.700025e-003
  A____1A 8_12              2.942102e+002  -1.138949e+003
  A 8_12A____2             -1.737797e+004
  A 8_12A____2                                           -1.568034e-001
  A____1A A_12             -2.195788e+001  -1.[illegible]4e+002
  A A_12A____2              2.769217e+005
  A A_12A____2                                           -3.635504e-001
  A____1A C_12             -1.356027e+003   9.905119e+002
  A C_12A____2              1.556051e+004
  A C_12A____2                                           4.199693e-002
  A____1A E_12              2.534693e+003  -8.123033e+002
  A E_12A____2             -1.426832e+004
  A E_12A____2                                           -2.083931e-002
  A____1A10_12             -3.480429e+003   5.651609e+002
  A10_12A____2              1.058648e+004
  A10_12A____2                                           1.425341e-002
  A____1A12_12             -1.427467e+002  -1.380378e+002
  A12_12A____2              2.771293e+004
  A12_12A____2                                           -4.480791e-002
  A____1A14_12             -1.835352e+002   1.407996e+002
  A14_12A____2              2.579641e+004
  A14_12A____2                                           2.457868e-002
C (2,2)
  A____2                    2.136131e+002
  A____2                   -1.697350e+001  -1.553794e+003
  A____2                    1.209292e+001   2.610945e+002
  A____2                    2.347825e+001   3.077445e+002
  A____2                    5.817348e+002   2.674856e+002
  A____2                   -4.503580e+002  -2.658903e+001
  A____2A 6__2              1.269424e+005  -4.880319e+004
  A 6__2                   -1.510270e+005
  A 6__2                                                 -2.295168e-003
  A____2A 8__2             -2.646082e+002   8.977345e+002
  A 8__2                    1.251412e+004
  A 8__2                                                  1.980824e-001
  A____2A A__2              9.729605e+000   7.751308e+001
  A A__2                    2.239360e+005
  A A__2                                                  7.274259e-001
  A____2A C__3              4.703554e+002  -1.343231e+003
  A C__2                   -5.811050e+004
  A C__2                                                 -3.365080e-002
  A____2A E__2             -3.314680e+003   6.780535e+002
  A E__2                    9.734108e+003
  A E__2                                                  2.002066e-002
  A____2A10__2             -2.573017e+002  -6.882284e+002
  A10__2                   -6.031784e+005
  A10__2                                                 -1.744483e-002
  A____2A12__2              4.967693e+001   6.967468e+001
  A12__2                   -2.319546e+004
  A12__2                                                  8.904087e-002
  A____2A14__2             -1.198030e+006   1.281169e+004
  A14__2                    1.237224e+006
  A14__2                                                  8.618294e-006
C ********************************* Beta Mode *********************************
C <BUS1><BUS2><BUS3><BUS4><   OHM        ><   milliH     ><   microF     >
C
C (1,1)
  B____1                    1.179990e+002
  B____1                   -1.817485e+001  -1.663768e+003
  B____1                    1.139488e+001   2.460232e+002
  B____1                    1.235152e+001   1.618993e+002
  B____1                    1.198652e+003   5.511484e+002
  B____1                   -1.989105e+002  -1.174363e+001
  B____1B 6__1              6.337633e+000   6.022723e+001
  B 6__1                    4.738628e+002
  B 6__1                                                  1.181832e+001
  B____1B 8__1              1.102904e+001   2.229382e+002
  B 8__1                    2.072046e+004
  B 8__1                                                  8.153085e-001
  B____1B A__1              1.013795e+001   7.765019e+001
  B A__1                    1.249981e+006
  B A__1                                                  7.261159e-001
  B____1B C__1              2.569920e+002   3.941270e+002
  B C__1                   -1.787782e+004
  B C__1                                                  1.139597e-001
  B____1B E__1             -1.620042e+002   2.269678e+002
  B E__1                    1.168918e+004
  B E__1                                                  8.943689e-002
  B____1B10__1              2.320907e+004   1.290952e+003
  B10__1                   -2.935271e+004
  B10__1                                                  1.945724e-003
  B____1B12__1              4.713920e+001   6.899455e+001
  B12__1                   -2.446764e+004
  B12__1                                                  8.993800e-002
  B____1B14__1             -2.288147e+006  -1.622627e+004
  B14__1                    2.321225e+006
  B14__1                                                 -3.060986e-006
C (1,2)
  B____1B____2             -6.862856e+003
  B____1B____2              5.790700e+001   5.300940e+003
  B____1B____2             -3.452911e+001  -7.455070e+002
  B____1B____2              1.627967e+001   2.133881e+002
  B____1B____2              3.760290e+003   1.729007e+003
  B____1B____2              2.107372e+004   1.244187e+003
  B____1B 6_12              3.509147e+004  -5.910586e+004
  B 6_12B____2             -1.309887e+005
  B 6_12B____2                                           -8.700025e-003
  B____1B 8_12              2.942102e+002  -1.138949e+003
  B 8_12B____2             -1.737797e+004
  B 8_12B____2                                           -1.568034e-001
```

```
  B____1B A_12                 2.195788e+001  -1.550764e+002
  B A_12B____2                 2.769217e+005
  B A_12B____2                                                -3.635504e-001
  B____1B C_12                -1.356027e+003   9.905119e+002
  B C_12B____2                 1.556051e+004
  B C_12B____2                                                 4.199693e-002
  B____1B E_12                 2.534693e+003  -8.123033e+002
  B E_12B____2                -1.426832e+004
  B E_12B____2                                                -2.083931e-002
  B____1B10_12                -3.480429e+003   5.651609e+002
  B10_12B____2                 1.058648e+004
  B10_12B____2                                                 1.425341e-002
  B____1B12_12                -1.427467e+002  -1.380378e+002
  B12_12B____2                 2.771293e+004
  B12_12B____2                                                -4.480791e-002
  B____1B14_12                -1.835352e+002   1.407996e+002
  B14_12B____2                 2.579641e+004
  B14_12B____2                                                 2.457868e-002
C (2,2)
  B____2                       2.136131e+002
  B____2                      -1.697350e+001  -1.553794e+003
  B____2                       1.209292e+001   2.610945e+002
  B____2                       2.347825e+001   3.077445e+002
  B____2                       9.817348e+002   2.674856e+002
  B____2                      -4.503580e+002  -2.658903e+001
  B____2B 6__2                 1.269424e+005  -4.880319e+004
  B 6__2                      -1.510270e+005
  B 6__2                                                      -2.295168e-003
  B____2B 8__2                -2.646082e+002   8.977345e+002
  B 8__2                       1.251412e+004
  B 8__2                                                      1.980824e-001
  B____2B A__2                 9.729605e+000   7.751308e+001
  B A__2                       2.239360e+005
  B A__2                                                      7.274259e-001
  B____2B C__2                 4.703554e+002  -1.343231e+003
  B C__2                      -5.811050e+004
  B C__2                                                     -3.365080e-002
  B____2B E__2                -3.314680e+003   6.780535e+002
  B E__2                       9.734108e+003
  B E__2                                                      2.002066e-002
  B____2B10__2                -2.573017e+002  -6.882284e+002
  B10__2                      -6.031784e+005
  B10__2                                                     -1.744483e-002
  B____2B12__2                 4.967693e+001   6.967468e+001
  B12__2                      -2.319546e+004
  B12__2                                                      8.904087e-002
  B____2B14__2                -1.198030e+006   1.281169e+004
  B14__2                       1.237224e+006
  B14__2                                                      8.618294e-006
$VINTAGE,0
C END FDNE
/SWITCH
C < n 1>< n 2>< Tclose ><Top/Tde ><   Ie   ><Vf/CLOP ><  type  >
C ********************************* Alpha Mode *********************************
  X0018ASZ__1A       .05        1.                                              0
C ********************************** Beta Mode **********************************
  X0018BSZ__1B       .05        1.                                              0
/SOURCE
C *********************************
C Norton equivalent current sources
C *********************************
C < n 1><> Ampl.  >< Freq.  ><Phase/T0><   A1   ><   T1   >< TSTART >< TSTOP  >
C ********************************* Alpha Mode *********************************
14IN__1A-12.01843989       60.    -86.21                          -1.        1.
14IN__2A-12.79744876       60.    -93.66                          -1.        1.
C ********************************** Beta Mode **********************************
14IN__1B-12.01843989       60.   -176.21                          -1.        1.
14IN__2B-12.79744876       60.   -183.66                          -1.        1.
/OUTPUT
  SZ__1ASZ__2ASZ__1BSZ__2B
BLANK BRANCH
BLANK SWITCH
BLANK SOURCE
BLANK OUTPUT
BLANK PLOT
BEGIN NEW DATA CASE
BLANK
```

B.4.2 Balanced Three-Phase to Ground Fault Case

```
BEGIN NEW DATA CASE
C --------------------------------------------------------
C Xin Nie
C Power Engineering Group
C Dept. of Electrical and Computer Engineering
C University of Alberta
C September 22, 2004
C --------------------------------------------------------
POWER FREQUENCY                    60.
$DUMMY, XYZ000
C  dT  >< Tmax >< Xopt >< Copt >
   2.E-5      .2
       1         1         0         0         0         0         0         0         1         0
C        1         2         3         4         5         6         7         8
C 345678901234567890123456789012345678901234567890123456789012345678901234567890
/BRANCH
C < n 1>< n 2><ref1><ref2>< R  >< L  >< C  >
C < n 1>< n 2><ref1><ref2>< R  >< A  >< B  ><Leng><><>0
C ********************************* Alpha Mode *********************************
C *************
C study zone
C *************
  SZ__1ASZTMPA             1.E-9                                                1
  SZ__1A                    200.  130.                                          0
  SZ__2A                    650.  250.                                          0
  SZ__1A                                8.                                      0
  IN__1A                   1200.  500.                                          0
  IN__2A                   2150.  380.                                          0
  X0018A                      2.                                                0
C HAWK conductor
-1SZTMPAIN__1A                2.  0.00                 -2 1
       12       4.090347064905766000E+02
  2.458905136002357 00E+03   1.006919164736511 00E+03   3.735522939590348 00E+03
  4.949108642839249 00E+03   1.809220174535260 00E+03   1.626389975440227 00E+03
  7.235902112013972 00E+02   9.980631571653958 00E+02   5.162740507080648 00E+02
  1.179555959812405 00E+04   6.126537567484807 00E+04   2.428839145698563 00E+06
  2.536132136180311 00E+00   3.678274032523600 00E+00   7.358252873864169 00E+00
  1.840262484773079 00E+01   3.580228449323820 00E+01   5.945998927028807 00E+01
  7.782171028008578 00E+01   1.297276129271710 00E+02   2.268054645514569 00E+02
  4.144249177986411 00E+03   2.155807371752076 00E+04   8.578379257482028 00E+05
       19       4.005194444611750 0000E-04
  1.569769744883085 00E-02   5.969318016973556 00E-02   7.657106329819022 00E-02
  1.230253109782413 00E-01   7.274362618176230 500E+00   1.156313267927832 00E+01
  1.517933952535450 00E+01   2.454600998092000 00E+01   2.175625245662915 00E+02
  4.751575573024700 00E+03   7.172772316756964 00E+03   2.154208551130008 00E+02
  1.927804409250900 00E+05   1.001302044569728 00E+05   6.676941419459103 00E+05
  4.498508068621796 00E+09  -4.592203111952144 00E+09   3.249801162280406 00E+09
 -3.157079140375575 00E+09
  5.460067998651540 00E+00   2.063829992047784 00E+01   2.694103636262369 00E+01
  4.366175083258754 00E+01   2.728891534101922 00E+03   3.969698516654513 00E+03
  5.156233595789232 00E+03   1.038658273690606 00E+04   9.267324595579616 00E+03
  1.124694586165610 00E+05   6.932342002009935 00E+04   4.989208395223323 00E+05
  4.157544530856369 00E+05   7.764223805854834 00E+05   1.439448374462287 00E+06
  4.025219577694497 00E+06   4.029244797262184 00E+06   4.381295402519039 00E+06
  4.385676697921551 00E+06
 1.00000000
 0.00000000
C DRAKE conductor
-1SZTMPASZ__2A                2.  0.00                 -2 1
       11       3.938973277035573 0000E+02
  2.271090787356759 00E+03   2.468738369802207 00E+03   2.767283308844692 00E+03
  9.272013278576356 00E+02   8.136523854897157 00E+02   4.843864533476302 00E+02
  6.006604547470770 00E+02   3.191230068771898 00E+02   1.474134983976601 00E+04
```

```
  5.35709730600259600E+04   2.76198969356044600E+06
  2.56520429418621500E+00   6.23867915240623700E+00   1.37315501948703500E+01
  2.44411566219346600E+01   3.88234599365499000E+01   5.22046486070647100E+01
  8.94985384307681800E+01   1.61248544547484800E+02   2.10024281549984200E+03
  3.24249725179149800E+04   1.15971094692565100E+06
       19        4.53915497928795800000E-04
  1.71275917005395300E-02   3.98691337390333500E-02   1.60864260029983000E-01
  6.89003618290678800E+00   8.78632925717092800E+00   1.18708400084055100E+01
  1.46650428128307100E+01   4.55132169053748200E+01   1.92681981165319200E+02
  2.21326495567518500E+03   1.11000476132130400E+04  -9.21432832870185500E+04
  2.81176544230162200E+05   1.67841941653263700E+04   1.22535036976167300E+06
 -3.16876240455654700E+07   2.93464632821155900E+07   3.77888719163484300E+07
 -3.68930729156446700E+07
  6.29920182790418300E+00   1.47535442461695500E+01   5.98892062931638000E+01
  2.46878104720658700E+03   3.24688225671564200E+03   4.20547568033697400E+03
  5.30166238127239600E+03   1.66566947044242800E+04   8.86571988667755600E+03
  4.65746128476995800E+04   1.13398074824504100E+05   4.43515608473281600E+05
  3.98417841600631700E+05   5.62068679963815700E+05   1.44682872910978600E+06
  2.87151967320063000E+06   2.87439119287383300E+06   3.65427188965292700E+06
  3.65792616154218300E+06
  1.00000000
  0.00000000
C HAWK conductor
-1SZ__2AIN__2A                2.  0.00                -2 1
       12        4.09034706498057660000E+02
  2.45898513600235700E+03   1.00691916473651100E+03   3.73552293959034800E+03
  4.94910864283924900E+03   1.80922017453526000E+03   1.62638997544022700E+03
  7.23598211201397200E+02   9.98061571653958900E+02   5.16274050708064800E+02
  1.17955959812405300E+04   6.12653756748480700E+04   2.42883914569856300E+06
  2.53613213618031100E+00   3.67827403252360000E+00   7.35825287306416900E+00
  1.84026248477307900E+01   3.58022844932382000E+01   5.94599892702880700E+01
  7.78217102800857800E+01   1.29727612927171000E+02   2.26805464551456900E+02
  4.14424917798641100E+03   2.15580737175207600E+04   8.57837925748202800E+05
       19        5.50554384535761800000E-04
  9.79465605855896600E-03   4.06212733123785000E-02   7.44854903179096900E-02
  4.00229870144877300E-02   6.56254564933406700E-02   7.79345690027440800E-02
  7.11151466287055800E-01   6.74005236856138300E+00   8.01620767358119800E+00
  1.17220670074905700E+01   3.50297130275519900E+01   7.05494675388751300E+01
  7.13600486817041200E+02   4.43545346148581900E+03   6.11781278301569400E+04
  6.79300571244501800E+04   7.49764722063926500E+05   2.36469459307105400E+07
 -2.45311009688204900E+07
  3.94587715068155200E+00   1.57035953406104900E+01   3.03393666019717600E+01
  1.67066122294504600E+01   2.60870054659298800E+01   3.32492107284800500E+01
  2.91988641610697900E+02   2.66461815651150000E+03   3.31965063922072700E+03
  4.56609698304445300E+03   7.47335672283102600E+03   6.69903629701688300E+03
  1.75902539791774200E+04   5.33686195521447800E+04   2.15476314218399300E+05
  3.13824776054083900E+05   9.37530861396163600E+05   1.82325148034484500E+06
  1.82507473182518700E+06
  1.00000000
  0.00000000
C *************
C surface layer
C *************
C DRAKE conductor
-1IN__1AA____1                2.  0.00                -2 1
        8        3.75023834778916070OE+002
  5.11823503089618290E+005  -7.72204001507163306E+004   5.38360813873814196E+005
 -1.31307871111864449E+006   1.41068547587384806E+006  -8.70567443420536826E+005
  3.97388861930764166E+005  -8.10869020300844631E+004
  1.09457825469020216E+004   8.94985384307681780E+001   5.22046486070647120E+001
  3.66517096081507320E+001   2.53973509773809386E+001   1.37315501948703500E+001
  6.00379679021483880E+000   2.66556099598264900E+000
       10        5.00972553527526500E-004
  1.62708639066469899E+008  -1.64315960011577190E+008   1.42142044554734910E+006
  1.08852614347084110E+005   6.41745286005793310E+004   1.15628088370959190E+004
  1.04986653831271150E+003   1.23014094562961110E+002   1.37442602299703910E+002
  2.26307971170046810E-001
  2.18519965186196890E+006   2.18301663522674100E+006   1.27972825220856490E+006
  4.18149118902648280E+005   2.57942363146516900E+005   1.12883452207787090E+005
  2.53652009135588810E+004   5.89469841003038530E+003   1.30053844190923900E+004
  2.22256160724067900E+001
  1.00000000
  0.00000000
C DRAKE conductor
-1IN__2AA____2                2.  0.00                -2 1
        8        3.93603297335005560OE+002
  1.05110454929315690E+005   2.41645471484145790E+002  -1.50171869550810123E+004
  6.34760464938527290E+004  -8.01358894723321280E+004   4.99703786877677192E+004
 -8.55372648222986750E+003   3.38808085731226920E+002
  1.09457825469020216E+004   8.94985384307681780E+001   5.22046486070647120E+001
  3.66517096081507320E+001   2.53973509773809388E+001   1.37315501948703500E+001
  6.00379679021483880E+000   2.66556099598264900E+000
       12        4.00426050961204500OE-004
  2.98214184532881620E+008  -3.00441936748123590E+008   1.94953735740764700E+006
  1.81190511239794810E+005   4.52870950130041160E+004   4.94167159533953670E+004
  1.85233960320056800E+003   3.63334026525288780E+002   5.80139940648757670E+001
  3.92689486018354610E+001   7.51715225293475430E+000   6.19059675115058230E-002
  3.19236050386884110E+006   3.18917133253630200E+006   1.85176609469088000E+006
  6.08443000187976200E+005   3.08295328076512320E+005   2.48181270816452710E+005
  4.07898477134318890E+004   1.59616192559506800E+004   9.92000427665665480E+003
  6.79698139061104300E+003   1.34382581690183410E+003   1.11009661252355690E+001
  1.00000000
  0.00000000
C HAWK conductor
-1IN__2AA____1                2.  0.00                -2 1
       12        4.09033489452554310OE+002
  2.42884613033530160E+006   6.12736636872050800E+004   1.17919069440423700E+004
  5.19565697093208770E+002   3.32364185903879790E+002   5.71643573674989780E+003
  3.26340041693269120E+004  -1.55865243163278440E+005   4.84121476804487230E+005
 -1.23847487601692700E+006   1.39441898554366310E+006  -5.05579166610773250E+005
  8.57837925748202830E+005   2.15580737175207610E+004   4.14424917798641130E+003
  2.26805464551456910E+002   1.29727612927171010E+002   7.78217102800857820E+001
  5.94599892702880690E+001   3.58022844932381970E+001   1.84026248477307920E+001
  7.35825287386416900E+000   3.67827403252359990E+000   2.53613213618031090E+000
       11        1.33553858593500000E-003
 -1.77483458695204400E+009   1.75077328882297990E+009   2.40238664000058411E+007
  2.17818936435455040E+004   1.45100081909931100E+004   5.99332332436985780E+002
  5.39057052699196900E+002   9.02910255312966310E-001   2.88828858700595600E-001
  1.82410892634117390E-001   6.36308228852859730E-002
  4.94559714131356872E+005   4.94065648664904580E+005   9.28595571451658850E+005
  9.37904960189008560E+004   5.82818523459502390E+004   7.04953628444685820E+003
  1.40902563620958700E+004   9.20243016657631900E+001   2.92663373528519910E+001
  1.78288418753010300E+001   6.18090131824526610E+000
  1.00000000
  0.00000000
C ********************************* Beta Mode *******************************
C *************
C study zone
C *************
  SZ__1BSZTMPB             1.E-9                                              1
  SZ__1B                    200.  130.                                        0
  SZ__2B                    650.  250.                                        0
  SZ__1B                                8.                                  0
  IN__1B                   1200.  500.                                        0
  IN__2B                   2150.  380.                                        0
  X0018B                      2.
C HAWK conductor
-1SZTMPBIN__1B                2.  0.00                -2 1
       12        4.09034706490576600000E+02
  2.45890513600235700E+03   1.00691916473651100E+03   3.73552293959034800E+03
  4.94910864283924900E+03   1.80922017453526000E+03   1.62638997544022700E+03
  7.23590211201397200E+02   9.98061571653958900E+02   5.16274050708064800E+02
  1.17955959812405300E+04   6.12653756748480700E+04   2.42883914569856300E+06
  2.53613213618031100E+00   3.67827403252360000E+00   7.35825287306416900E+00
  1.84026248477307900E+01   3.58022844932382000E+01   5.94599892702880700E+01
  7.78217102800857800E+01   1.29727612927171000E+02   2.26805464551456900E+02
  4.14424917798641100E+03   2.15580737175207600E+04   8.57837925748202800E+05
       19        4.00519444461175000000E-04
  1.56976974488308500E-02   5.96931801697356600E-02   7.65710632981902200E-02
  1.23025310978241300E-01   7.74362618176230500E+00   1.15631326792783200E+01
  1.51791335253545000E+01   2.45460099809200000E+01   2.17562524566291500E+02
  4.75157557302470000E+03   7.17277231675696400E+03   2.15420855113000800E+02
  1.92780440925090000E+05   1.00130204456972800E+05   6.67694141945910300E+05
  4.49850806862179600E+09  -4.59220311195214400E+09   3.24980116228040600E+09
 -3.15707914037557500E+09
  5.46006799865154000E+00   2.06382999204778400E+01   2.69410363626236900E+01
```

```
  4.366175083258754000E+01   2.728891534101922000E+03   3.969698516654513000E+03
  5.156233595578923200E+03   1.038658273690606000E+04   9.267324595579616000E+03
  1.124694586165610000E+05   6.932342002009935000E+04   4.989208395223323000E+05
  4.157544530856369000E+05   7.764223805854834000E+05   1.439448374462287000E+06
  4.025219577684497000E+06   4.029244797262184000E+06   4.381295402519039000E+06
  4.385676697921551000E+06
 1.00000000
 0.00000000
C DRAKE conductor
-1SZTMPBSZ__2B                2.  0.00                -2 1
    11        3.938973277035573000E+02
  2.271090787356759000E+03   2.468738369802207000E+03   2.767283308844692000E+03
  9.272013278576356000E+02   8.136523854897157000E+02   4.843866453347630200E+02
  6.006604547477077000E+02   3.191230068771898000E+02   1.474134983976601000E+04
  5.357097306002596000E+04   2.741989693560646000E+06
  2.565204294186215000E+00   6.238679152406237000E+00   1.373155019487035000E+01
  2.444115662193466000E+01   3.882345993654990000E+01   5.220464860706471000E+01
  8.949853843076818000E+01   1.612465445474848000E+02   6.100242815499842000E+03
  2.242497251791498000E+04   1.159710946925651000E+06
     19        4.539154979279580000E-04
  1.712759170053953000E+02   3.986913373903335000E-02   1.608642600299830000E-01
  6.890036182906788000E+00   8.786329257170928000E+00   1.187084000840551000E+01
  1.465504281283071000E+01   4.551321695053748200E+01   1.998819811653192000E+02
  2.213264955675185000E+03   1.110004761321304000E+04   9.144328328701855000E+04
  2.817765442301622000E+05   1.678419416532637000E+04   1.225350369761673000E+06
 -3.168762404556547000E+07   2.934646328211559000E+07   3.778887191634843000E+07
 -3.689307291564467000E+07
  6.298201827904183000E+00   1.475354424616955000E+01   5.988920629316388000E+01
  2.468781047206587000E+03   3.246882256715642000E+03   4.205475680336974000E+03
  5.301662381272396000E+03   1.665669470442428000E+04   8.865719886677556000E+03
  4.657461284769958000E+04   1.133980748245041000E+05   4.435156084732816000E+05
  3.984178416006317000E+05   5.620686799638157000E+05   1.446828729109786000E+06
  2.871519673200630000E+06   2.874391192873833000E+06   3.654271889652527000E+06
  3.657926161542183000E+06
 1.00000000
 0.00000000
C HAWK conductor
-1SZ__2BIN__2B                2.  0.00                -2 1
    12        4.090347064905766000E+02
  2.458905136002357000E+03   1.006919164736511000E+03   3.735522939590348000E+03
  4.949108642839249000E+03   1.809220174535260000E+03   1.626389975440227000E+03
  7.235902112013972000E+02   9.980615716539589000E+02   5.162740507080648000E+02
  1.179559598124053000E+04   6.126537567484807000E+04   2.428839145698563000E+06
  2.536132136180311000E+00   3.678274032523600000E+00   7.358252873864169000E+00
  1.840262484773079000E+01   3.580228449323820000E+01   5.945998927028807000E+01
  7.782171028008578000E+01   1.297276129271710000E+02   2.268054645514569000E+02
  4.144249177986411000E+03   2.155807317175207600E+04   8.578379257748202800E+05
     19        5.505543845357618000E-04
  9.794656058558596000E-03   4.062127331237850000E-02   7.448549031790969000E-02
  4.002298701448773000E-02   6.562545649334067000E-02   7.793456900274408000E-02
  7.111514662870558000E-01   6.740052368561383000E+00   8.016207673581198000E+00
  1.172206700749057000E+01   3.502971302755199000E+01   7.054946753887513000E+01
  7.136004868170412000E+02   4.435453461485819000E+03   6.117812783015694000E+04
  6.793005712445018000E+04   7.497647220639265000E+05   2.364694593071054000E+07
 -2.453110096882049000E+07
  3.945877150681552000E+00   1.570359534061049000E+01   3.033936660197176000E+01
  1.670661222945046000E+01   2.608700546592988000E+01   3.324921072848005000E+01
  2.919886416106979000E+02   2.664618156511500000E+03   3.319650639220727000E+03
  4.566096983044453000E+03   7.473356722831026000E+03   6.699036297016883000E+03
  1.759025397917744200E+04   5.336861955214478000E+04   2.154763142183993000E+05
  3.138247765054083900E+05   9.375308613961636000E+05   1.823251480344845000E+06
  1.825074731825187000E+06
 1.00000000
 0.00000000
C *************
C surface layer
C *************
C DRAKE conductor
-1IN__1BB____1                2.  0.00                -2 1
     8        3.750238347789160700E+002
  5.118235030896182 9E+005  -7.722040015021633 0E+004   5.383608138738141 9E+005
 -1.313078711186444 9E+006   1.410685475873848 0E+006  -8.705674434205368 2E+005
  3.973888619307641 6E+005  -8.108690203008463 1E+004
```

```
  1.094578254690202 1E+004   8.949853843076817 8E+001   5.220464860706471 2E+001
  3.665170960815073 2E+001   2.539735097738093 8E+001   1.373155019487035 0E+001
  6.003796790214838 8E+000   2.665560959826490 E+000
     10        5.009725535275262500E-004
  1.627086390644658 5E+008  -1.643159600115771 9E+008   1.421420445547349 1E+006
  1.088526143470841 1E+005   6.417452860057933 1E+004   1.156280883705951 9E+004
  1.049866538312715 0E+003   1.230140945629611 1E+002   1.374426022997039 1E+002
  2.263079711700468 1E-001
  2.185199651861968 9E+006   2.183016653522674 0E+006   1.279728252208564 9E+006
  4.181491189026482 0E+005   2.579423631465169 0E+005   1.128834522077870 9E+005
  2.536520091355888 1E+004   5.894698410030385 3E+003   1.300538441909239 0E+004
  2.222561607240679 0E+001
 1.00000000
 0.00000000
C DRAKE conductor
-1IN__2BB____2                2.  0.00                -2 1
     8        3.939032973350055600E+002
  1.051104549293156 9E+005   2.416454714841457 9E+002  -1.501718695581012 3E+004
  6.347604649385272 9E+004  -8.013588947233212 8E+004   4.997037868767719 2E+004
 -8.553726482229867 5E+003   3.388080857312269 2E+002
  1.094578254690202 1E+004   8.949853843076817 8E+001   5.220464860706471 2E+001
  3.665170960815073 2E+001   2.539735097738093 8E+001   1.373155019487035 0E+001
  6.003796790214838 8E+000   2.665560959826490 E+000
     12        4.004260509612045000E-004
  2.982141845328816 2E+008  -3.004419367481235 9E+008   1.949537357407647 0E+006
  1.811905112397948 1E+005   4.528709501300411 6E+004   4.941671595339536 7E+004
  1.852339603200568 0E+003   3.633340265252887 8E+002   5.801399406487577 6E+001
  3.926894860183546 1E+001   7.517152252934754 3E+000   6.190596751150582 3E-002
  3.192360503868841 1E+006   3.189171332536302 0E+006   1.851766094690880 0E+006
  6.084430081879762 0E+005   3.082953280765123 2E+005   2.481812708164527 1E+005
  4.078984771343188 9E+004   1.596161925595068 0E+004   9.920004276656654 8E+003
  6.796981390611043 0E+003   1.343825816901834 1E+003   1.110096612523556 9E+001
 1.00000000
 0.00000000
C HAWK conductor
-1IN__2BB____1                2.  0.00                -2 1
    12        4.090334894525543100E+002
  2.428846130335301 6E+006   6.127366368720508 0E+004   1.179190694404237 0E+004
  5.195656970932087 7E+002   3.323641859038797 9E+002   5.716435736749897 8E+003
  3.263400416932691 2E+004  -1.558652431632784 4E+005   4.841214768044872 3E+005
 -1.238474876016927 0E+006   1.394418985543663 1E+006  -5.055791666107732 5E+005
  8.578379257482028 3E+005   2.155803717520761 E+004   4.144249177986411 3E+003
  2.268054645514569 1E+002   1.297276129271710 1E+002   7.782171028008578 2E+001
  5.945998927028806 9E+001   3.580228449323819 7E+001   1.840262484773079 2E+001
  7.358252873864168 8E+000   3.678274032523599 9E+000   2.536132136180310 9E+000
     11        1.335553858593508 0000E-003
 -1.774834586952044 0E+009   1.750773288822979 9E+009   2.402386640005841 1E+007
  2.178189364335041 E+004   1.451000819099311 0E+004   5.993323324369857 8E+002
  5.390570562699196 0E+002   9.029102553129663 1E+001   2.888288587005956 0E-001
  1.824108926341173 9E-001   6.363082288528597 3E-002
  4.945597143135687 2E+005   4.940656486649045 8E+005   5.285955714516588 5E+005
  9.379049601890085 6E+004   5.828185234595023 9E+004   7.049536284446858 2E+003
  1.409025636209587 0E+004   9.202430166576319 0E+001   2.926633735285191 9E+001
  1.782884187530103 0E+001   6.180901318245266 1E+000
 1.00000000
 0.00000000
C ********************************* Alpha Mode *******************************
C *************
C deep region
C *************
C BEGIN FDNE
$VINTAGE,1
C <BUS1><BUS2><BUS3><BUS4><   OHM        ><   milliH      ><   microF     >
C
C (1,1)
  A____1                      1.179990e+002
  A____1                     -1.817485e+001  -1.663768e+003
  A____1                      1.139488e+001   2.460232e+002
  A____1                      1.235152e+001   1.618993e+002
  A____1                      1.198652e+003   5.511484e+002
  A____1                     -1.989105e+002  -1.174363e+001
  A____1A 6__1                6.337633e+000   6.022723e+001
  A 6__1                      4.738628e+002
```

```
A 6__1
A____1A 8__1          1.102904e+001   2.229382e+002
A 8__1                2.072046e+004
A 8__1
A____1A A__1          1.013795e+001   7.765019e+001
A A__1                1.249981e+006
A A__1
A____1A C__1          2.569920e+002   3.941270e+002
A C__1               -1.787782e+004
A C__1
A____1A E__1         -1.620042e+002   2.269678e+002
A E__1                1.168918e+004
A E__1
A____1A10__1          2.320907e+004   1.290952e+003
A10__1               -2.935271e+004
A10__1
A____1A12__1          4.713920e+001   6.899455e+001
A12__1               -2.446764e+004
A12__1
A____1A14__1         -2.288147e+006  -1.622627e+004
A14__1                2.321225e+006
A14__1
C (1,2)
A____4A____2         -6.862856e+003
A____1A____2          5.790700e+001   5.300940e+003
A____1A____2         -3.452911e+001  -7.455070e+002
A____1A____2          1.627967e+001   2.133881e+002
A____1A____2          3.760290e+003   1.729007e+003
A____1A____2          2.107372e+004   1.244187e+003
A____1A 6_12          3.509147e+004  -5.910586e+004
A 6_12A____2         -1.309887e+005
A 6_12A____2
A____1A 8_12          2.942102e+002  -1.138949e+003
A 8_12A____2         -1.737797e+004
A 8_12A____2
A____1A A_12         -2.195788e+001  -1.550764e+002
A A_12A____2          2.769217e+005
A A_12A____2
A____1A C_12         -1.356027e+003   9.905119e+002
A C_12A____2          1.556051e+004
A C_12A____2
A____1A E_12          2.534693e+003  -8.123033e+002
A E_12A____2         -1.426832e+004
A E_12A____2
A____1A10_12         -3.480429e+003   5.651609e+002
A10_12A____2          1.058648e+004
A10_12A____2
A____1A12_12         -1.427467e+002  -1.380378e+002
A12_12A____2          2.771293e+004
A12_12A____2
A____1A14_12         -1.835352e+002   1.407996e+002
A14_12A____2          2.579841e+004
A14_12A____2
C (2,2)
A____2                2.136131e+002
A____2               -1.697350e+001  -1.553794e+003
A____2                1.209292e+001   2.610945e+002
A____2                2.347825e+001   3.077445e+002
A____2                5.817348e+002   2.674856e+002
A____2               -4.503580e+002  -2.658903e+001
A____2A 6__2          1.269424e+005  -4.880319e+004
A 6__2               -1.510270e+005
A 6__2
A____2A 8__2         -2.646082e+002   8.977345e+002
A 8__2                1.251412e+004
A 8__2
A____2A A__2          9.729603e+000   7.751308e+001
A A__2                2.239360e+005
A A__2
A____2A C__2          4.703554e+002  -1.343231e+003
A C__2               -5.811050e+004
A C__2
A____2A E__2         -3.314680e+003   6.780535e+002
A E__2                9.734108e+003
A E__2
A____2A10__2         -2.573017e+002  -6.882284e+002
A10__2               -6.031784e+005
A10__2
A____2A12__2          4.967693e+001   6.967468e+001
A12__2               -2.319546e+004
A12__2
A____2A14__2         -1.198030e+006   1.281169e+004
A14__2                1.237224e+006
A14__2
C ********************************* Beta Mode *********************************
C <BUS1><BUS2><BUS3><BUS4><   OHM        ><   milliH     ><   microF     >
C
C (1,1)
B____1                1.179990e+002
B____1               -1.817485e+001  -1.663768e+003
B____1                1.139488e+001   2.460232e+002
B____1                1.235152e+001   1.618993e+002
B____1                1.198652e+003   5.511484e+002
B____1               -1.989105e+002  -1.174363e+001
B____1B 6__1          6.337633e+000   6.022723e+001
B 6__1                4.738628e+002
B 6__1
B____1B 8__1          1.102904e+001   2.229382e+002
B 8__1                2.072046e+004
B 8__1
B____1B A__1          1.013795e+001   7.765019e+001
B A__1                1.249981e+006
B A__1
B____1B C__1          2.569920e+002   3.941270e+002
B C__1               -1.787782e+004
B C__1
B____1B E__1         -1.620042e+002   2.269678e+002
B E__1                1.168918e+004
B E__1
B____1B10__1          2.320907e+004   1.290952e+003
B10__1               -2.935271e+004
B10__1
B____1B12__1          4.713920e+001   6.899455e+001
B12__1               -2.446764e+004
B12__1
B____1B14__1         -2.288147e+006  -1.622627e+004
B14__1                2.321225e+006
B14__1
C (1,2)
B____1B____2         -6.862856e+003
B____1B____2          5.790700e+001   5.300940e+003
B____1B____2         -3.452911e+001  -7.455070e+002
B____1B____2          1.627967e+001   2.133881e+002
B____1B____2          3.760290e+003   1.729007e+003
B____1B____2          2.107372e+004   1.244187e+003
B____1B 6_12          3.509147e+004  -5.910586e+004
B 6_12B____2         -1.309887e+005
B 6_12B____2
B____1B 8_12          2.942102e+002  -1.138949e+003
B 8_12B____2         -1.737797e+004
B 8_12B____2
B____1B A_12         -2.195788e+001  -1.550764e+002
B A_12B____2          2.769217e+005
B A_12B____2
B____1B C_12         -1.356027e+003   9.905119e+002
B C_12B____2          1.556051e+004
B C_12B____2
B____1B E_12          2.534693e+003  -8.123033e+002
B E_12B____2         -1.426832e+004
B E_12B____2
B____1B10_12         -3.480429e+003   5.651609e+002
B10_12B____2          1.058648e+004
B10_12B____2
B____1B12_12         -1.427467e+002  -1.380378e+002
B12_12B____2          2.771293e+004
B12_12B____2
```

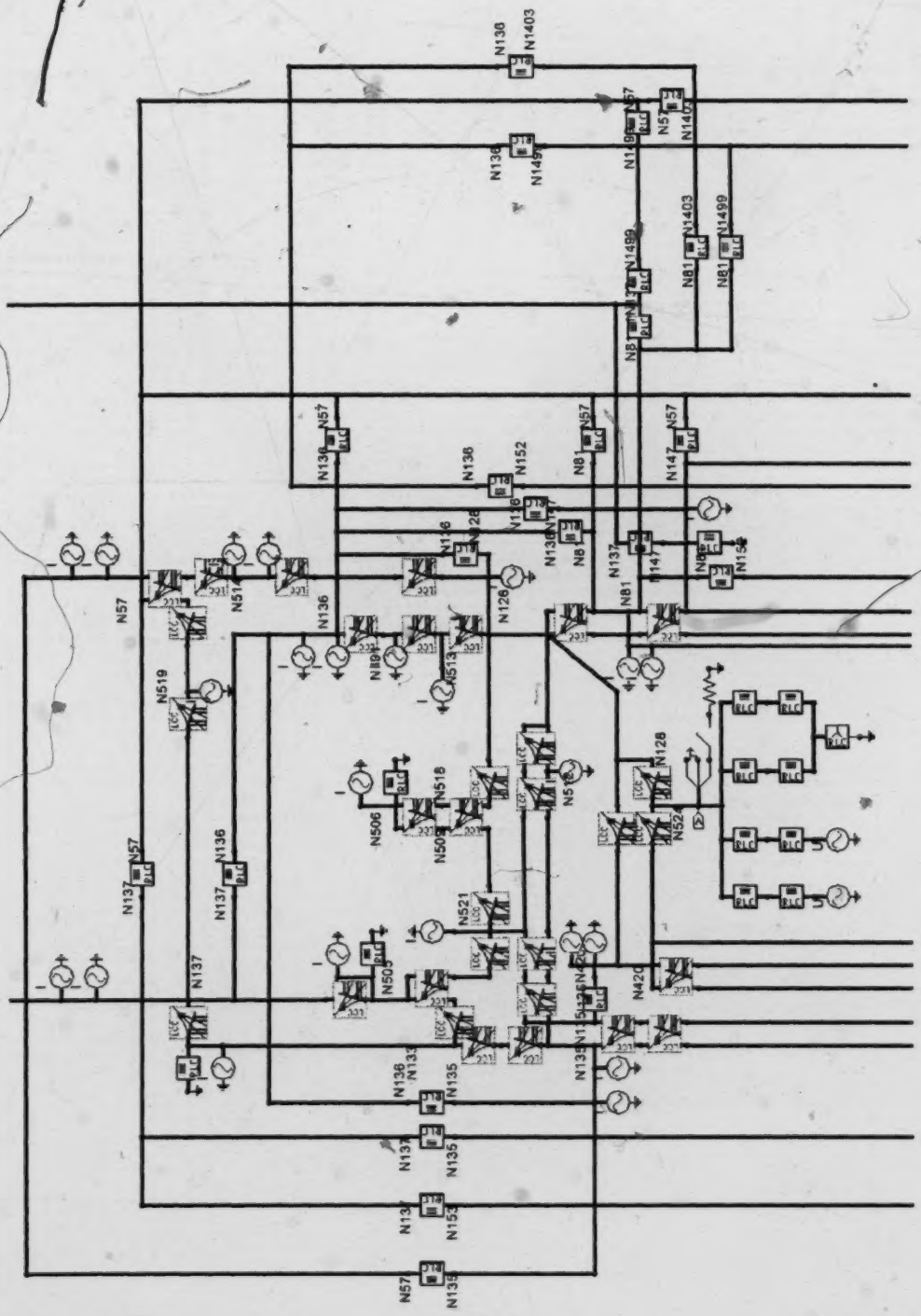
```
B____1B14_12          -1.835352e+002  1.407996e+002
B14_12B____2           2.579641e+004
B14_12B____2
C (2,2)                                2.457868e-002
B____2
B____2                 2.136131e+002
B____2                -1.697350e+001  -1.553794e+003
B____2                 1.209292e+001   2.610945e+002
B____2                 2.347825e+001   3.077445e+002
B____2                -5.817348e+002  -2.674856e+002
B____2B 6__2          -4.503590e+002  -2.658903e+001
B____2                -1.269424e+005  -4.880319e+004
B____2                -1.510270e+005
B____2                                -2.295168e-003
B____2B 8__2          -2.646082e+002   8.977345e+002
B____2                 1.251412e+004
B____2B 8__2                           1.980824e-001
B____2B A__2           9.729605e+000   7.751308e+001
B____2                 2.239360e+005
B____2                                7.274259e-001
B____2B C__2           4.703554e+002  -1.343231e+003
B____2                -5.811050e+004
B____2                                -3.365080e-002
B____2B E__2          -3.314680e+003   6.708535e+002
B____2                 8.734108e+003
B____2B10__2          -2.573017e+002  -6.882284e+002   2.002066e-002
B10__2                -6.031784e+005
B10__2                                -1.744482e-002
B____2B12__2           4.967693e+001   6.967468e+001
B12__2                -2.319546e+004
B12__2                                 8.904087e-002
```

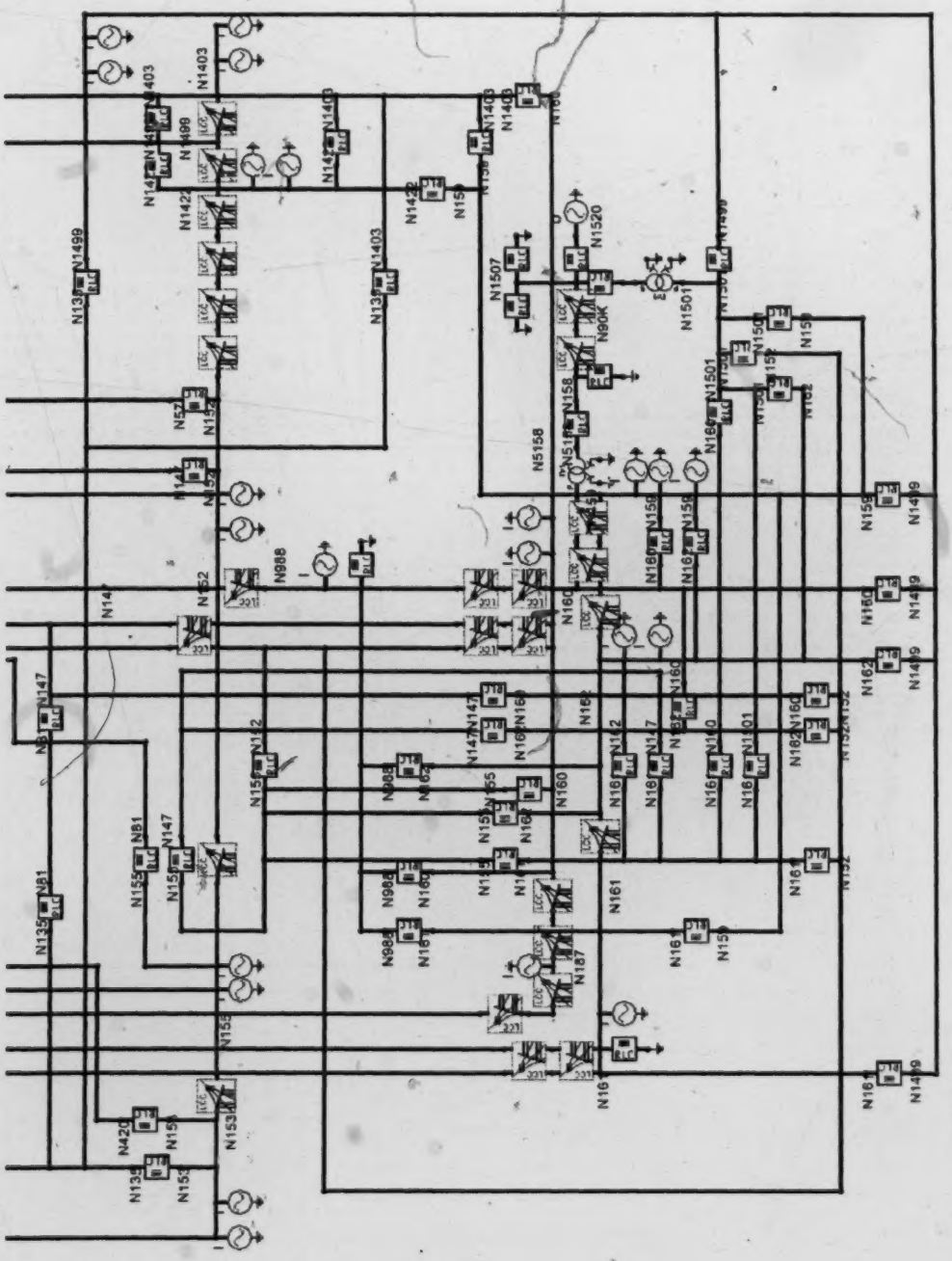
```
B____2B14__2          -1.198030e+006  1.281169e+004
B14__2                 1.237224e+006
B14__2                                 8.618294e-006
$VINTAGE,0
C END FDNE
/SWITCH
C <  n 1><  n 2>< Tclose ><Top/Tde ><   Ie    ><VI/CLOP >< type >
C ***************************************** Alpha Mode ******************************************
 X0018AS2__2A          .05          .15
C ***************************************** Beta Mode *******************************************
 X0018BS2__2B          .05          .15
/SOURCE
C <  n 1><2>< Ampl.  ><  Freq.  ><Phase/TO><   A1   ><   T1   >< TSTART >< TSTOP  >
C ***************************************** Alpha Mode ******************************************
C Norton equivalent current sources
C ***************************************** Beta Mode *******************************************
14IN__1A-12.01843989          60.     -86.21                                        -1.       1.
14IN__2A-12.79744876          60.     -93.66                                        -1.       1.
E ***************************************** Beta Mode *******************************************
14IN__1B-12.01843989          60.    -176.21                                        -1.       1.
14IN__2B-12.79744876          60.    -183.66                                        -1.       1.
/OUTPUT
 S2_1AS2__2AS2__1BS2__2B
BLANK BRANCH
BLANK SWITCH
BLANK SOURCE
BLANK OUTPUT
BLANK PLOT
BEGIN NEW DATA CASE
BLANK
```

C AIES Area 50 Backbone

C.1 AIES Area 50 Backbone Diagram in ATP

The diagram in ATPDraw is split into two figures, which are the upper and lower parts.


N138
N1403
N1499
N81
N57
N136
N152
N147
N126
N519
N513
N518
N506
N521
N128
N52
N420
N135
N133
N153
N571



C.2 PSS/E Procedures in Obtaining Equivalents for Area 50 Backbone

```
                    POWER TECHNOLOGIES INCORPORATED

          4000 BUS POWER SYSTEM SIMULATOR--PSS/E-26.1

 INITIATED AT LOAD FLOW ENTRY POINT ON SUN, JUL 31 2005  20:42

 Executing activity read

read
 ENTER INPUT FILE NAME (0 TO EXIT, 1 FOR TERMINAL): d:\aies\aies.raw

 ENTER IC, SBASE

 ENTER TWO LINE HEADING

 ENTER BUS DATA

 ENTER LOAD DATA

 ENTER GENERATOR DATA

 MESSAGES FOR MACHINE 1  AT BUS  4185 [SCOTF GT13.800]:
  WARNING: MACHINE IS OFF-LINE--BUS TYPE CODE IS  1

 MESSAGES FOR MACHINE 1  AT BUS  4187 [CECGT   18.000]:
  WARNING: MACHINE IS OFF-LINE--BUS TYPE CODE IS  1

 MESSAGES FOR MACHINE 2  AT BUS  7165 [SCOT4 ST13.800]:
  WARNING: MACHINE IS OFF-LINE--BUS TYPE CODE IS  1

 ENTER BRANCH DATA

 ENTER TRANSFORMER ADJUSTMENT DATA

 ENTER AREA INTERCHANGE DATA

 ENTER TWO-TERMINAL DC LINE DATA

 ENTER SWITCHED SHUNT DATA

 ENTER TRANSFORMER IMPEDANCE CORRECTION DATA

 ENTER MULTI-TERMINAL DC LINE DATA

 ENTER MULTI-SECTION LINE DATA

 ENTER ZONE NAME DATA

 ENTER INTER-AREA TRANSFER DATA

 ENTER OWNER NAME DATA

 ENTER FACTS CONTROL DEVICE DATA

 ACTIVITY?
 Executing activity fnsl

fnsl

 ORDERING NETWORK
 DIAGONALS =  1686  OFF-DIAGONALS =  2466  MAX SIZE =  3694

 ENTER ITERATION NUMBER FOR VAR LIMITS
  0 FOR IMMEDIATELY, -1 TO IGNORE COMPLETELY:

 ITER   DELTAP   BUS     DELTAQ   BUS     DELTA/V/   BUS   DELTAANG   BUS
   0    0.0237(  229)    0.0548( 3069)     0.05482( 3069)   0.01289( 4264)
   1    0.0010(   69)    0.2816( 3069)     0.04936( 3069)   0.01318( 4264)
   2    0.0011(  111)    0.0134( 3069)     0.00278( 3069)   0.00065(19124)
   3    0.0005(  299)    0.0020(  825)     0.00030(  379)   0.00033(  299)
   4    0.0005(  299)    0.0010(  924)

 REACHED TOLERANCE IN   4 ITERATIONS

 LARGEST MISMATCH:    0.03 MW   -0.12 MVAR     0.12 MVA-BUS   825 [CGETAP  138.00]
 SYSTEM TOTAL ABSOLUTE MISMATCH:               2.75 MVA

 SWING BUS SUMMARY:
    BUS X--- NAME ---X     PGEN     PMAX     PMIN     QGEN     QMAX     QMIN
  1520 WSCC GEN500.00     177.0 158154.7-87991.2    -139.5  63541.1-48593.3

 ACTIVITY?
 Executing activity eeqv,area

eeqv,area

 USER SPECIFIES SUBSYSTEM TO BE EQUIVALENCED
  ENTER UP TO 20 AREA NUMBERS
4,6,13,17,18,19,20,21,22,23,24,25,26,27,28,29,30
  ENTER UP TO 20 AREA NUMBERS
31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49
  ENTER UP TO 20 AREA NUMBERS
52,53,54,55,56,57,60,88,89,91,92,97
  ENTER UP TO 20 AREA NUMBERS


 ENTER 1 TO RETAIN AREA BOUNDARY BUSES:

 ENTER 1 TO RETAIN ZONE BOUNDARY BUSES:

 ENTER 1 TO SUPPRESS EQUIVALENCING OF PHASE SHIFTERS:

 ENTER 1 TO RETAIN BUSES CONTROLLED BY REMOTE GENERATION OR SWITCHED SHUNT:

 ENTER MINIMUM GENERATION FOR RETAINING GENERATOR BUSES
    (CARRIAGE RETURN TO KEEP ALL ON-LINE GENERATOR BUSES): 1000

 ENTER 1 TO RETAIN EXISTING BRANCHES BETWEEN RETAINED BUSES: 1
 1492 RADIAL AND TWO POINT BUSES EQUIVALENCED

 ENTER BRANCH THRESHOLD TOLERANCE:
 DIAGONALS =   155  OFF-DIAGONALS =  742  MAX SIZE =  1045

 ENTER 1 TO NET LOAD AND SHUNT AT RETAINED BUSES:

 ACTIVITY?
 Executing activity eeqv

eeqv

 USER SPECIFIES SUBSYSTEM TO BE EQUIVALENCED
  ENTER UP TO 20 BUS NUMBERS
36,40,41,63,87,89,90,99,109,117,118,129,130,131,132,145,146,148
  ENTER UP TO 20 BUS NUMBERS
151,154,156,163,165,202,207,208,281,338,342,345,348,350,374,422,423,424
  ENTER UP TO 20 BUS NUMBERS
491,492,495,496,516,542,545,805,806,943,1260,1280,1484,1489,1497,2144,3150
  ENTER UP TO 20 BUS NUMBERS
3187,4158,4187,5159,5161,5505,5506,5526,6135,6144,6506,6526,10226,10228,10312
  ENTER UP TO 20 BUS NUMBERS
10422,11228,11312,18403,25053,25054,25123,25124,25135,25136,25193,25194
  ENTER UP TO 20 BUS NUMBERS
25213,25214,138,1226,1228,1312,1348,19226
  ENTER UP TO 20 BUS NUMBERS


 ENTER 1 TO RETAIN AREA BOUNDARY BUSES:

 ENTER 1 TO RETAIN ZONE BOUNDARY BUSES:
```

```
ENTER 1 TO RETAIN BUSES CONTROLLED BY REMOTE GENERATION OR SWITCHED SHUNT:

ENTER MINIMUM GENERATION FOR RETAINING GENERATOR BUSES
  (CARRIAGE RETURN TO KEEP ALL ON-LINE GENERATOR BUSES): 1000

ENTER 1 TO RETAIN EXISTING BRANCHES BETWEEN RETAINED BUSES: 1
  54 RADIAL AND TWO POINT BUSES EQUIVALENCED

ENTER BRANCH THRESHOLD TOLERANCE:

MATRIX TOO BIG IN ORDR AT ROW    30--INCREASING MATRIX GROWTH FACTOR TO  2.25

MATRIX TOO BIG IN ORDR AT ROW    30--INCREASING MATRIX GROWTH FACTOR TO  2.50
DIAGONALS =    66  OFF-DIAGONALS =   414  MAX SIZE =   605

ENTER 1 TO NET LOAD AND SHUNT AT RETAINED BUSES:

ACTIVITY?
```

C.3 Area 50 Backbone ATP Data Files

C.3.1 Full Model

```
BEGIN NEW DATA CASE
C -----------------------------------------------------------
C Xin Nie
C Power Engineering Group
C Dept. of Electrical and Computer Engineering
C University of Alberta
C July 19, 2005
C -----------------------------------------------------------
C Alberta Interconnected Electric System 240kV Area 50 Backbone
C
C
$DUMMY, XY2000
C  dT  >< Tmax >< Xopt >< Copt >
   1.E-5      .15      60.      60.
     500         1       0        0       0       0       0       1       0
C        1         2         3         4         5         6         7         8
C 34567890123456789012345678901234567890123456789012345678901234567890123456789012345678
/BRANCH
C < n 1>< n 2><ref1><ref2>< R  >< L  >< C  >
C < n 1>< n 2><ref1><ref2>< R  >< A  >< B  ><Leng><>><0
  N1507AN1520A              5.574.175                                          0
  N1507BN1520B              5.574.175                                          0
  N1507CN1520C              5.574.175                                          0
  N1422AN1403A           1645.3 3347.                                          0
  N1422BN1403B           1645.3 3347.                                          0
  N1422CN1403C           1645.3 3347.                                          0
  N137A N57A                11.87 99.23                                       0
  N137B N57B                11.87 99.23                                       0
  N137C N57C                11.87 99.23                                       0
  N81A  N137A               73.45623.32                                       0
  N81B  N137B               73.45623.32                                       0
  N81C  N137C               73.45623.32                                       0
  N135A N137A               55.94275.21                                       0
  N135B N137B               55.94275.21                                       0
  N135C N137C               55.94275.21                                       0
  N137A N136A                 66.292.98                                       0
  N137B N136B                 66.292.98                                       0
  N137C N136C                 66.292.98                                       0
  N153A N137A            1581.54387.3                                         0
  N153B N137B            1581.54387.3                                         0
  N153C N137C            1581.54387.3                                         0
  N137A N1499A           1041.3242.4                                          0
  N137B N1499B           1041.3242.4                                          0
  N137C N1499C           1041.3242.4                                          0
  N81A  N57A                162.3465.31                                       0
  N81B  N57B                162.3465.31                                       0
  N81C  N57C                162.3465.31                                       0
  N135A N57A                38.15171.02                                       0
  N135B N57B                38.15171.02                                       0
  N135C N57C                38.15171.02                                       0
  N136A N57A                 9.21 50.75                                       0
  N136B N57B                 9.21 50.75                                       0
  N136C N57C                 9.21 50.75                                       0
  N147A N57A             2105.33105.4                                         0
  N147B N57B             2105.33105.4                                         0
  N147C N57C             2105.33105.4                                         0
  N152A N57A              3076.3887.3                                         0
  N152B N57B              3076.3887.3                                         0
  N152C N57C              3076.3887.3                                         0
  N1403AN57A              244.51054.3                                         0
  N1403BN57B              244.51054.3                                         0
  N1403CN57C              244.51054.3                                         0
  N1499AN57A              135.6499.92                                         0
  N1499BN57B              135.6499.92                                         0
  N1499CN57C              135.6499.92                                         0
  N81A  N136A               34.12255.58                                       0
  N81B  N136B               34.12255.58                                       0
  N81C  N136C               34.12255.58                                       0
  N135A N136A            286.43773.28                                         0
  N135B N136B            286.43773.28                                         0
  N135C N136C            286.43773.28                                         0
  N147A N136A              733.98 1790.                                       0
  N147B N136B              733.98 1790.                                       0
  N147C N136C              733.98 1790.                                       0
  N152A N136A            1141.32287.8                                         0
  N152B N136B            1141.32287.8                                         0
  N152C N136C            1141.32287.8                                         0
  N1403AN136A             855.062615.2                                        0
  N1403BN136B             855.062615.2                                        0
  N1403CN136C             855.062615.2                                        0
  N1499AN136A             186.84768.65                                        0
  N1499BN136B             186.84768.65                                        0
  N1499CN136C             186.84768.65                                        0
  N506A                                    184.03                            0
  N506B                                    184.03                            0
  N506C                                    184.03                            0
       N133A             2438.616583.                                        0
       N133B             2438.616583.                                        0
       N133C             2438.616583.                                        0
  N505A                                    197.92                            0
  N505B                                    197.92                            0
  N505C                                    197.92                            0
  N135A N1403A           980.243957.6                                         0
  N135B N1403B           980.243957.6                                         0
  N135C N1403C           980.243957.6                                         0
  N152A N147A               3.97 59.55                                        0
  N152B N147B               3.97 59.55                                        0
  N152C N147C               3.97 59.55                                        0
  N128A N136A                .56  3.17                                       0
  N128B N136B                .56  3.17                                       0
  N128C N136C                .56  3.17                                       0
  N135A N81A             295.611145.6                                         0
  N135B N81B             295.611145.6                                         0
  N135C N81C             295.611145.6                                         0
  N153A N135A             84.72283.61                                         0
  N153B N135B             84.72283.61                                         0
  N153C N135C             84.72283.61                                         0
  N135A N420A            305.11836.69                                         0
  N135B N420B            305.11836.69                                         0
  N135C N420C            305.11836.69                                         0
  N135A N1499A           485.391860.5                                         0
  N135B N1499B           485.391860.5                                         0
  N135C N1499C           485.391860.5                                         0
  N153A N420A               84.371.24                                         0
```

| | | |

```
   N5158AN158A                .275 24.08                                          0
   N5158BN158B                .275 24.08                                          0
   N5158CN158C                .275 24.08                                          0
   N524A X0376A              .38621.715                                           0
   N524B X0376B              .38621.715                                           0
   N524C X0376C              .38621.715                                           0
   N524A X0378A               .3821.882                                           0
   N524B X0378B               .3821.882                                           0
   N524C X0378C               .3821.882                                           0
   N524A X0380A             1.733871.654                                          0
   N524B X0380B             1.733871.654                                          0
   N524C X0380C             1.733871.654                                          0
   N524A X0382A             1.693472.236                                          0
   N524B X0382B             1.693472.236                                          0
   N524C X0382C             1.693472.236                                          0
   X0376AX0384A              .245432.287                                          0
   X0376BX0384B              .245432.287                                          0
   X0376CX0384C              .245432.287                                          0
   X0378AX0386A              .245432.287                                          0
   X0378BX0386B              .245432.287                                          0
   X0378CX0386C              .245432.287                                          0
   X1071A                   14615.10167.                                          0
   X1071B                   14615.10167.                                          0
   X1071C                   14615.10167.                                          0
   X0380AX1071A             1.733871.654                                          0
   X0380BX1071B             1.733871.654                                          0
   X0380CX1071C             1.733871.654                                          0
   X0382AX1071A             1.693472.236                                          0
   X0382BX1071B             1.693472.236                                          0
   X0382CX1071C             1.693472.236                                          0
C 1201-1,2 - (1201L), CROWSNEST (4 bundle)
$INCLUDE, E:\ATP\LCC\1201_12.1ib, N158A#, N158B#, N158C#, N90KA#, N90KB# $$
  , N90KC#
C 1201-3 - (1201L), 666.9 (4 bundle)
$INCLUDE, E:\ATP\LCC\1201_3.1ib, N90KA#, N90KB#, N90KC#, N1507A, N1507B $$
  , N1507C
C Segment 1 : 936-1 - (936L), 937-1 - (937L), 1590 SDC TYPE 9
$INCLUDE, E:\ATP\LCC\936937_1.1ib, N160A#, N160B#, N160C#, N160A#, N160B# $$
  , N160C#, X0014A, X0014B, X0014C, X0015A, X0015B, X0015C
C Segment 2 : 936-2 - (936L), 937-2 - (937L), CURLEW
$INCLUDE, E:\ATP\LCC\923924_2.1ib, X0014A, X0014B, X0014C, X0015A, X0015B $$
  , X0015C, N159A#, N159B#, N159C#, N159A#, N159B#, N159C#
C 9L20, 397.5 (similar to IBIS)
$INCLUDE, E:\ATP\LCC\9L20.1ib, N1422A, N1422B, N1422C, N1499A, N1499B $$
  , N1499C
C 9L27 - (9L27), IBIS
$INCLUDE, E:\ATP\LCC\9L27.1ib, N1499A, N1499B, N1499C, N1403A, N1403B $$
  , N1403C
C 918-6,7a,8a - (918L), HAWK and 477 SDC 5005
$INCLUDE, E:\ATP\LCC\918_678.1ib, X0401A, X0401B, X0401C, N187A#, N187B# $$
  , N187C#
C Segment 1 : 906-134 - (906L), 477 SDC EC, 928-134 - (928L), SDC 5005
$INCLUDE, E:\ATP\LCC\906928_1.1ib, X0026A, X0026B, X0026C, X0027A, X0027B $$
  , X0027C, N155A#, N155B#, N155C#, N155A#, N155B#, N155C#
C Segment 2 : 906-2,5 - (906L), 928-2,5 - (928L), CURLEW, (CARDINAL/SD merged)
$INCLUDE, E:\ATP\LCC\906928_2.1ib, N161A#, N161B#, N161C#, N161A#, N161B# $$
  , N161C#, X0026A, X0026B, X0026C, X0027A, X0027B, X0027C
C Segment 1 : 925-12 - (925L), SDC 5005, 929-123 - (929L), SDC 5005
$INCLUDE, E:\ATP\LCC\925929_1.1ib, X0034A, X0034B, X0034C, X0035A, X0035B $$
  , X0035C, N152A#, N152B#, N152C#, N152A#, N152B#, N152C#
C Segment 2 : 925-345 - (925L), 929-4,5 - (929L), CURLEW
$INCLUDE, E:\ATP\LCC\925929_2.1ib, N160A#, N160B#, N160C#, N160A#, N160B# $$
  , N160C#, X0034A, X0034B, X0034C, X0035A, X0035B, X0035C
C 901-1^5 - (901L), SDC 5005 and HAWK
$INCLUDE, E:\ATP\LCC\901_1_5.1ib, N988A#, N988B#, N988C#, N152A#, N152B# $$
  , N152C#
C 901-6 - (901L), HAWK
$INCLUDE, E:\ATP\LCC\901_6.1ib, X0044A, X0044B, X0044C, N988A#, N988B# $$
  , N988C#
C 901-7 - (901L), CURLEW
$INCLUDE, E:\ATP\LCC\901_7.1ib, N160A#, N160B#, N160C#, X0044A, X0044B $$
  , X0044C
C 900 - (900L), HAWK                                                              0
$INCLUDE, E:\ATP\LCC\900.1ib, N155A#, N155B#, N155C#, N152A#, N152B#, N152C#      0
C 995 - (995L), HAWK and SDC 5005 (19.97km), N150 (68S) and N995 ignored          0
$INCLUDE, E:\ATP\LCC\995.1ib, N153A#, N153B#, N153C#, N155A#, N155B#, N155C#      0
C 912-123 - (912L), SDC 5005                                                      0
$INCLUDE, E:\ATP\LCC\912_123.1ib, N152A#, N152B#, N152C#, X0053A, X0053B $$       0
  , X0053C                                                                        0
C 917-12 - (917L), CURLEW                                                         0
$INCLUDE, E:\ATP\LCC\917.1ib, N162A#, N162B#, N162C#, N160A#, N160B#, N160C#      0
C 916 - (916L), DRAKE                                                             0
$INCLUDE, E:\ATP\LCC\916.1ib, N161A#, N161B#, N161C#, N162A#, N162B#, N162C#      0
C 912-4 - (912L), IBIS                                                            0
$INCLUDE, E:\ATP\LCC\912_4.1ib, X0053A, X0053B, X0053C, X0059A, X0059B $$         0
  , X0059C                                                                        0
C 918-1^5 - (918L), HAWK, 3.08km CURLEW and  SDC 6101 was merged                  0
$INCLUDE, E:\ATP\LCC\918_1_5.1ib, X0401A, X0401B, X0401C, N155A#, N155B# $$       0
  , N155C#                                                                        0
C 912-5 - (912L), 477 SDC 6101                                                    0
$INCLUDE, E:\ATP\LCC\912_5.1ib, X0059A, X0059B, X0059C, X0063A, X0063B $$         0
  , X0063C                                                                        0
C 912-6 - (912L), IBIS                                                            0
$INCLUDE, E:\ATP\LCC\912_6.1ib, X0063A, X0063B, X0063C, N1422A, N1422B $$         0
  , N1422C                                                                        0
C Segment 2 : 922-34789 - (922L), 926-34789 - (926L), SDC 5005 and HAWK           0
$INCLUDE, E:\ATP\LCC\922926_2.1ib, N155A#, N155B#, N155C#, N155A#, N155B# $$      0
  , N155C#, X0070A, X0070B, X0070C, X0071A, X0071B, X0071C                        0
C Segment 1 : 922-1256 - (922L), 926-1256 - (926L), CURLEW                        0
$INCLUDE, E:\ATP\LCC\922926_1.1ib, X0070A, X0070B, X0070C, X0071A, X0071B $$      0
  , X0071C, N135A#, N135B#, N135C#, N135A#, N135B#, N135C#                        0
C 903 - (903L), 190 - (190L), HAWK                                                0
$INCLUDE, E:\ATP\LCC\903190.1ib, N155A#, N155B#, N155C#, N155A#, N155B# $$        0
  , N155C#, N420A#, N420B#, N420C#, N420A#, N420B#, N420C#                        0
C Segment 3 : 910-34 - (910L), IBIS, 914-456 - (914L), 447 SDC 6101               0
$INCLUDE, E:\ATP\LCC\910914_3.1ib, N152A#, N152B#, N152C#, N152A#, N152B# $$      0
  , N152C#, X0681A, X0681B, X0681C, N147A#, N147B#, N147C#                        0
C Segment 2 : 910-2 - (910L), 447 SDC 5005, 914-3 - (914L), 447 SDC 6101          0
$INCLUDE, E:\ATP\LCC\910914_2.1ib, X0681A, X0681B, X0681C, N147A#, N147B# $$      0
  , N147C#, X0443A, X0443B, X0443C, N81A##, N81B##, N81C##                        0
C Segment 1 : 910-1 - (910L), 447 SDC 5005, 914-12 - (914L), 447 SDC 6101         0
$INCLUDE, E:\ATP\LCC\910914_1.1ib, X0443A, X0443B, X0443C, N81A##, N81B## $$      0
  , N81C##, N128A#, N128B#, N128C#, N128A#, N128B#, N128C#                        0
C Segment 2 : 908-2^5 - (908L), 909-2^6 - (909L), 477 SDC TYPE 6.4                0
$INCLUDE, E:\ATP\LCC\908909_2.1ib, X0092A, X0092B, X0092C, X0093A, X0093B $$      0
  , X0093C, N521A#, N521B#, N521C#, X0537A, X0537B, X0537C                        0
C Segment 1 : 908-1 - (908L), 909-1 - (909L), CURLEW                              0
$INCLUDE, E:\ATP\LCC\908909_1.1ib, N135A#, N135B#, N135C#, N135A#, N135B# $$      0
  , N135C#, X0092A, X0092B, X0092C, X0093A, X0093B, X0093C                        0
C Segment 3 : 908-6 - (908L), 909-7 - (909L), 477 SDC TYPE 6.4                    0
$INCLUDE, E:\ATP\LCC\908909_3.1ib, N521A#, N521B#, N521C#, X0537A, X0537B $$      0
  , X0537C, X0102A, X0102B, X0102C, N512A#, N512B#, N512C#                        0
C Segment 4 : 908-7,8 - (908L), 909-8^12 - (909L), 477 SDC TYPE 6.4               0
$INCLUDE, E:\ATP\LCC\908909_4.1ib, X0102A, X0102B, X0102C, N512A#, N512B# $$      0
  , N512C#, N128A#, N128B#, N128C#, N128A#, N128B#, N128C#                        0
C 1202 - (1202L), 3 bundle 1590 SDC TYPE 7/9, maybe 500kV line                    0
$INCLUDE, E:\ATP\LCC\1202.1ib, N420A#, N420B#, N420C#, N128A#, N128B# $$          0
  , N128C#                                                                        0
C 1203 - (1203L), 3 bundle DRAKE, maybe 500kV line                                0
$INCLUDE, E:\ATP\LCC\1203.1ib, N420A#, N420B#, N420C#, N524A#, N524B# $$          0
  , N524C#                                                                        0
C 1209 - (1209L), 3 bundle 1590 SDC TYPE 7/9, TRILLIUM, maybe 500kV line          0
$INCLUDE, E:\ATP\LCC\1209.1ib, N524A#, N524B#, N524C#, N128A#, N128B# $$          0
  , N128C#                                                                        0
C 902-34 - (902L), HAWK and SDC 5005                                              0
$INCLUDE, E:\ATP\LCC\902_34.1ib, X0114A, X0114B, X0114C, N133A#, N133B# $$        0
  , N133C#                                                                        0
C 902-1256 - (902L), CURLEW                                                       0
$INCLUDE, E:\ATP\LCC\902_1256.1ib, N135A#, N135B#, N135C#, X0114A, X0114B $$      0
  , X0114C                                                                        0
C 904-134 - (904L), CURLEW                                                        0
$INCLUDE, E:\ATP\LCC\904_134.1ib, X0118A, X0118B, X0118C, X0475A, X0475B $$       0
  , X0475C, N505A#, N505B#, N505C#, N505A#, N505B#, N505C#                        0
C 904-2 - (904L), DRAKE                                                           0
$INCLUDE, E:\ATP\LCC\904_2.1ib, N133A#, N133B#, N133C#, X0118A, X0118B $$         0
  , X0118C                                                                        0
```

```
C 904-567 - (904L), DRAKE
$INCLUDE, E:\ATP\LCC\904_567.11b, X0475A, X0475B, X0475C, N521A#, N521B# $$
 , N521C#
C 904-8,9,10 - (904L), DRAKE
$INCLUDE, E:\ATP\LCC\904_8910.11b, N521A#, N521B#, N521C#, X0531A, X0531B $$
 X0531C
C 904-11-14 - (904L), CURLEW
$INCLUDE, E:\ATP\LCC\904_11_4.11b, X0531A, X0531B, X0531C, X0129A, X0129B $$
 X0129C, N509A#, N509B#, N509C#, N518A#, N518B#, N518C#
C 904-15-17 - (904L), 447 SDC
$INCLUDE, E:\ATP\LCC\904_15_7.11b, X0129A, X0129B, X0129C, N128A#, N128B# $$
 N128C#
C 904-18-21 - (904L), 447 SDC
$INCLUDE, E:\ATP\LCC\904_18_2.11b, N126A#, N126B#, N128C#, N513A#, N513B# $$
C 904-22-24 - (904L), 447 SDC
$INCLUDE, E:\ATP\LCC\904_22_4.11b, N513A#, N513B#, N513C#, N891A#, N891B# $$
 N891C#
C 904-25-26 - (904L), 447 SDC
$INCLUDE, E:\ATP\LCC\904_25_6.11b, N891A#, N891B#, N891C#, N136A#, N136B# $$
 N136C#
C 908-9-12 - (908L), 447 SDC 6.4
$INCLUDE, E:\ATP\LCC\908_9_12.11b, N128A#, N128B#, N128C#, N136A#, N136B# $$
 N136C#
C 905 - (905L), DRAKE
$INCLUDE, E:\ATP\LCC\905.11b, N133A#, N133B#, N133C#, N137A#, N137B#, N137C#
C 920-12 - (920L), DRAKE
$INCLUDE, E:\ATP\LCC\920_12.11b, N137A#, N137B#, N137C#, N519A#, N519B# $$
 N519C#
C 240BA2 & 240BA3, 2500 MCM U/G CABLE, 1590 SDC assumed
$INCLUDE, E:\ATP\LCC\240BA2_3.11b, N509A#, N509B#, N509C#, N518A#, N518B# $$
 N518C#, N506A#, N506B#, N506C#, N506A#, N506B#, N506C#
C 920-3 - (920L), DRAKE
$INCLUDE, E:\ATP\LCC\920_3.11b, N519A#, N519B#, N519C#, X0151A, X0151B $$
 X0151C
C 920-4 - (920L), HORNBILL, 921-4 - (921L), HORNBILL
$INCLUDE, E:\ATP\LCC\920921.11b, X0151A, X0151B, X0151C, X0153A, X0153B $$
 X0153C, N57A#, N57B#, N57C#, N57A#, N57B#, N57C#
C 921-123 - (921L), CURLEW
$INCLUDE, E:\ATP\LCC\921_123.11b, N514A#, N514B#, N514C#, X0153A, X0153B $$
 X0153C
C 915L-1,2 (915L), CURLEW
$INCLUDE, E:\ATP\LCC\915.11b, N136A#, N136B#, N136C#, N514A#, N514B#, N514C#
C 930-12 - (930L), CURLEW
$INCLUDE, E:\ATP\LCC\930.11b, N505A#, N505B#, N505C#, N137A#, N137B#, N137C#
C 932-1,2 - (932L), 447 SDC 5005
$INCLUDE, E:\ATP\LCC\932_12.11b, N107A#, N107B#, N107C#, X0163A, X0163B $$
 X0163C
C 932-3 - (932L), CURLEW
$INCLUDE, E:\ATP\LCC\932_3.11b, X0163A, X0163B, X0163C, N160A#, N160B# $$
 N160C#
/SOURCE
C <><Ampl. ><Freq. ><Phase/T0><   A1   ><   T1   >< TSTART >< TSTOP >
14N159A          1.E-20  60.                                  -1.      10.
14N159B          1.E-20  60.                                  -1.      10.
14N159C          1.E-20  60.                                  -1.      10.
14N1520A 0         416.41  60.       -120.                    -1.      -1.
14N1520B 0         416.41  60.       120.                     -1.      -1.
14N1520C 0         416.41  60.       0.                       -1.      -1.
14N137A -1          .252   60.       -82.72                   -1.      -1.
14N137B -1          .252   60.       -202.72                  -1.      -1.
14N137C -1          .252   60.       37.28                    -1.      -1.
14N137A -1          .6163  60.       150.89                   -1.      -1.
14N137B -1          .6163  60.       30.89                    -1.      -1.
14N137C -1          .6163  60.       270.89                   -1.      -1.
14N57A  -1         1.9733  60.       -80.77                   -1.      -1.
14N57B  -1         1.9733  60.       -200.77                  -1.      -1.
14N57C  -1         1.9733  60.       39.23                    -1.      -1.
14N57A  -1         1.777   60.       114.5                    -1.      -1.
14N57B  -1         1.777   60.       -5.5                     -1.      -1.

14N57C  -1         1.777   60.       234.5                    -1.      -1.
14N519A -1          .153   60.       171.19                   -1.      -1.
14N519B -1          .153   60.       51.19                    -1.      -1.
14N519C -1          .153   60.       291.19                   -1.      -1.
14N514A -1          .133   60.       -75.01                   -1.      -1.
14N514B -1          .133   60.       -195.01                  -1.      -1.
14N514C -1          .133   60.       44.99                    -1.      -1.
14N514A -1         1.209   60.       -16.68                   -1.      -1.
14N514B -1         1.209   60.       -103.32                  -1.      -1.
14N514C -1         1.209   60.       136.68                   -1.      -1.
14N136A -1          .347   60.       -75.62                   -1.      -1.
14N136B -1          .347   60.       -195.62                  -1.      -1.
14N136C -1          .347   60.       44.38                    -1.      -1.
14N136A -1          .9516  60.       154.14                   -1.      -1.
14N136B -1          .9516  60.       34.14                    -1.      -1.
14N136C -1          .9516  60.       274.14                   -1.      -1.
14N891A -1          .0755  60.       171.52                   -1.      -1.
14N891B -1          .0755  60.       51.52                    -1.      -1.
14N891C -1          .0755  60.       291.52                   -1.      -1.
14N506A -1          .7193  60.       160.46                   -1.      -1.
14N506B -1          .7193  60.       40.46                    -1.      -1.
14N506C -1          .7193  60.       280.46                   -1.      -1.
14N505A -1          .5534  60.       166.22                   -1.      -1.
14N505B -1          .5534  60.       46.22                    -1.      -1.
14N505C -1          .5534  60.       286.22                   -1.      -1.
14N133A -1          .7857  60.       24.76                    -1.      -1.
14N133B -1          .7857  60.       -95.24                   -1.      -1.
14N133C -1          .7857  60.       144.76                   -1.      -1.
14N521A -1          .1503  60.       173.85                   -1.      -1.
14N521B -1          .1503  60.       53.85                    -1.      -1.
14N521C -1          .1503  60.       293.85                   -1.      -1.
14N513A -1          .2626  60.       166.27                   -1.      -1.
14N513B -1          .2626  60.       46.27                    -1.      -1.
14N513C -1          .2626  60.       286.27                   -1.      -1.
14N128A -1          .046   60.       -75.45                   -1.      -1.
14N128B -1          .046   60.       -195.45                  -1.      -1.
14N128C -1          .046   60.       44.55                    -1.      -1.
14N512A -1          .1423  60.       169.24                   -1.      -1.
14N512B -1          .1423  60.       49.24                    -1.      -1.
14N512C -1          .1423  60.       289.24                   -1.      -1.
14N135A -1          4.3    60.       -80.67                   -1.      -1.
14N135B -1          4.3    60.       -200.67                  -1.      -1.
14N135C -1          4.3    60.       39.33                    -1.      -1.
14N135A -1         5.538   60.       59.1                     -1.      -1.
14N135B -1         5.538   60.       -60.9                    -1.      -1.
14N135C -1         5.538   60.       179.1                    -1.      -1.
14N187A -1         .01737  60.       137.43                   -1.      -1.
14N187B -1         .01737  60.       17.43                    -1.      -1.
14N187C -1         .01737  60.       257.43                   -1.      -1.
14N420A -1         2.093   60.       21.41                    -1.      -1.
14N420B -1         2.093   60.       141.41                   -1.      -1.
14N420C -1         2.093   60.       -58.59                   -1.      -1.
14N420A -1         .1066   60.       -71.28                   -1.      -1.
14N420B -1         .1066   60.       -191.28                  -1.      -1.
14N420C -1         .1066   60.       48.72                    -1.      -1.
14N81A  -1         .3994   60.       152.88                   -1.      -1.
14N81B  -1         .3994   60.       32.88                    -1.      -1.
14N81C  -1         .3994   60.       272.88                   -1.      -1.
14N81A  -1          .107   60.       -86.15                   -1.      -1.
14N81B  -1          .107   60.       -206.15                  -1.      -1.
14N81C  -1          .107   60.       33.85                    -1.      -1.
14N153A -1         .2909   60.       -162.31                  -1.      -1.
14N153B -1         .2909   60.       -282.31                  -1.      -1.
14N153C -1         .2909   60.       -42.31                   -1.      -1.
14N153A -1          .1803  60.       -84.83                   -1.      -1.
14N153B -1          .1803  60.       -204.83                  -1.      -1.
14N153C -1          .1803  60.       35.17                    -1.      -1.
14N155A -1         .3235   60.       158.68                   -1.      -1.
14N155B -1         .3235   60.       38.68                    -1.      -1.
14N155C -1         .3235   60.       278.68                   -1.      -1.
14N155A -1          .253   60.       -84.47                   -1.      -1.
14N155B -1          .253   60.       -204.47                  -1.      -1.
14N155C -1          .253   60.       35.53                    -1.      -1.
```

```
14N147A -1      .322        60.     162.84                  -1.       1.
14N147B -1      .322        60.      42.84                  -1.       1.
14N147C -1      .322        60.     282.84                  -1.       1.
14N152A -1     .4045        60.     158.28                  -1.       1.
14N152B -1     .4045        60.      38.28                  -1.       1.
14N152C -1     .4045        60.     278.28                  -1.       1.
14N152A -1      .198        60.      -87.9                  -1.       1.
14N152B -1      .198        60.     -207.9                  -1.       1.
14N152C -1      .198        60.       32.1                  -1.       1.
14N1501A      1.E-20        60.                             -1.      10.
18              .468X0895A
14N1501B      1.E-20        60.                             -1.      10.
18              .468X0895B
14N1501C      1.E-20        60.                             -1.      10.
18              .468X0895C
14N1422A-1      .1848       60.     144.06                  -1.       1.
14N1422B-1      .1848       60.      24.06                  -1.       1.
14N1422C-1      .1848       60.     264.06                  -1.       1.
14N1422A-1       .068       60.      -93.92                 -1.       1.
14N1422B-1       .068       60.     -213.92                 -1.       1.
14N1422C-1       .068       60.      26.08                  -1.       1.
14N1403A-1      .6883       60.     134.84                  -1.       1.
14N1403B-1      .6883       60.      14.84                  -1.       1.
14N1403C-1      .6883       60.     254.84                  -1.       1.
14N1403A-1       .462       60.      -97.78                 -1.       1.
14N1403B-1       .462       60.     -217.78                 -1.       1.
14N1403C-1       .462       60.      22.22                  -1.       1.
14N1499A-1      .9033       60.      72.09                  -1.       1.
14N1499B-1      .9033       60.     -47.91                  -1.       1.
14N1499C-1      .9033       60.     192.09                  -1.       1.
14N1499A-1       1.14       60.      -95.1                  -1.       1.
14N1499B-1       1.14       60.     -215.1                  -1.       1.
14N1499C-1       1.14       60.       24.9                  -1.       1.
14N988A -1      .2742       60.      153.5                  -1.       1.
14N988B -1      .2742       60.       33.5                  -1.       1.
14N988C -1      .2742       60.      273.5                  -1.       1.
14N160A -1     1.6443       60.     142.02                  -1.       1.
14N160B -1     1.6443       60.      22.02                  -1.       1.
14N160C -1     1.6443       60.     262.02                  -1.       1.
14N160A -1      1.222       60.      -95.32                 -1.       1.
14N160B -1      1.222       60.     -215.32                 -1.       1.
14N160C -1      1.222       60.      24.68                  -1.       1.
14N161A -1     1.6625       60.      147.1                  -1.       1.
14N161B -1     1.6625       60.       27.1                  -1.       1.
14N161C -1     1.6625       60.      267.1                  -1.       1.
14N162A -1      .8402       60.     145.08                  -1.       1.
14N162B -1      .8402       60.      25.08                  -1.       1.
14N162C -1      .8402       60.     265.08                  -1.       1.
14N162A -1       .733       60.      -96.92                 -1.       1.
14N162B -1       .733       60.     -216.92                 -1.       1.
14N162C -1       .733       60.      23.08                  -1.       1.
14N159A -1      .1061       60.      83.93                  -1.       1.
14N159B -1      .1061       60.     -36.07                  -1.       1.
14N159C -1      .1061       60.     203.93                  -1.       1.
14N159A -1      1.016       60.     -98.66                  -1.       1.
14N159B -1      1.016       60.     -218.66                 -1.       1.
14N159C -1      1.016       60.      21.34                  -1.       1.
14N159A -1      .8152       60.     110.73                  -1.       1.
14N159B -1      .8152       60.      -9.27                  -1.       1.
14N159C -1      .8152       60.     230.73                  -1.       1.
14X0384A 0      207.7       60.       28.5                  -1.       1.
14X0384B 0      207.7       60.      -91.5                  -1.       1.
14X0384C 0      207.7       60.      148.5                  -1.       1.
14X0386A 0      207.7       60.       29.9                  -1.       1.
14X0386B 0      207.7       60.      -90.1                  -1.       1.
14X0386C 0      207.7       60.      149.9                  -1.       1.
/OUTPUT
  N524A
BLANK BRANCH
BLANK SWITCH
BLANK SOURCE
BLANK OUTPUT
BLANK PLOT
```

```
BEGIN NEW DATA CASE
BLANK
```

C.3.2 Robust TLNE Model

```
BEGIN NEW DATA CASE
C ------------------------------------------------------------
C Xin Nie
C Power Engineering Group
C Dept. of Electrical and Computer Engineering
C University of Alberta
C August 25, 2005
C ------------------------------------------------------------
C Alberta Interconnected Electric System 240kV Area 50 Backbone
C
C
$DUMMY, XYZ000
C  dT  >< Tmax >< Xopt >< Copt >
   2.E-5     .15
       1       1       0       0       0       0       0       0       1       0
C      1       2       3       4       5       6       7       8
C 345678901234567890123456789012345678901234567890123456789012345678901234567890
/BRANCH
C < n 1>< n 2><ref1><ref2>< R  >< L  >< C  >
C < n 1>< n 2><ref1><ref2>< R  >< A  >< B  ><Leng><>0
C ************
C study zone
C ************
C ********************************** Alpha Mode ********************************
  NN524AX0010A              .386  57.6                                        0
  NN524AX0012A               .38 58.04                                        0
  NN524AX0014A            1.7338190.07                                        0
  NN524AX0016A            1.6934 191.6                                        0
  X0010AX0018A             .2454 85.64                                        0
  X0012AX0020A             .2454 85.64                                        0
  X0047A                  14615.26969.                                        0
  X0014AX0047A            1.7338190.07                                        0
  X0016AX0047A            1.6934 191.6                                        0
C Fault resistance is 10ohm
  X0030A                    10.                                               0
C ********************************** Beta Mode *********************************
  NN524BX0010B              .386  57.6                                        0
  NN524BX0012B               .38 58.04                                        0
  NN524BX0014B            1.7338190.07                                        0
  NN524BX0016B            1.6934 191.6                                        0
  X0010BX0018B             .2454 85.64                                        0
  X0012BX0020B             .2454 85.64                                        0
  X0047B                  14615.26969.                                        0
  X0014BX0047B            1.7338190.07                                        0
  X0016BX0047B            1.6934 191.6                                        0
C Fault resistance is 10ohm
  X0030B                    10.                                               0
C ************
C surface layer
C ************
C ********************************** Alpha Mode ********************************
C 1202 - (1202L), 3 bundle 1590 SDC TYPE 7/9
-1A____2A____1                  2.  0.00                -2 1
     6          2.2862969763617426 00E+002
   1.0930875266152098E+004   3.7816977451926283E+002   7.2244742454667050E+002
   4.1950782676470499E+002   2.2555335480015390E+002   1.5690211858979180E+002
   6.1738261875301032E+003   2.4272139132891308E+001   4.2730997978570935E+000
   1.1090996364568724E+000   2.8428629943126288E-001   1.0228943543026825E-001
      7          2.3079513625135739 00E-004-
   1.1105422223173680E+006   3.8383326840361638E+004   1.3390360725888700E+006
   3.2319994632951602E+002   2.5118915993528501E+002   7.7028931875867855E+001
   4.4701114841863907E+000
   1.2399131976713850E+006   5.8232972072919679E+005   6.8208797858970356E+004
   7.7755873710647822E+004   5.0009371993671411E+004   1.6343106099887820E+004
   9.6476396587423778E+002
```

```
  1.00000000
  0.00000000
C 1203 - (1203L), 3 bundle DRAKE
-1A____2NN524A                2.  0.00              -2 1
        6       2.35229445783656l000E+002
   3.95059139542088889E+004   8.01974122254ll199E+002   1.47251423773319l6E+003
   8.22016760521079l6E+002   4.44135227292783l8E+002   2.77941793750997l6E+002
   2.66809360982867l9E+004   4.49211411400484473E+001   7.78322785913622l1E+000
   1.86937975998527l1E+000   4.27694837032311l7E-001   1.11872929265317l7E-001
        5       6.10422368881328920lE-005
   2.41516096734507l0E+006  -2.82306923430l8622E+004   7.44494515752063l8E+002
   7.27454626704187l8E+002   6.14644360804775l2E+001
   2.44114964668l7860E+006   4.57971485248l8365E+006   1.38918887734267l1E+005
   1.23176856272740l0E+005   1.12879665693264l1E+004
  1.00000000
  0.00000000
C 1209 - (1209L), 3 bundle 1590 SDC TYPE 7/9, TRILLIUM
-1A____1NN524A                2.  0.00              -2 1
        6       2.28629697636174260lE+002
   1.09308752661520l8E+004   3.78169774519262l3E+002   7.22447424546670l0E+002
   4.19507826764704l9E+002   2.25553354800l5390E+002   1.56902118589791l0E+002
   6.17382618753010l2E+003   2.42721391328913l8E+001   4.27309979785709l5E+000
   1.10909963645687l4E+000   2.84286299431262l8E-001   1.02289435430268l5E-001
        7       2.24555200482619610lE-004
   9.98874107240689l0E+006   1.10086473725561l0E+005   1.35200927653972l0E+003
   2.89287468770128l8E+002   2.38000457034562l1E+002   7.06678707780972l0E+001
   4.35059666741596l3E+000
   1.19442152421977l0E+006   8.71736957821651l3E+005   6.95475731557374l7E+004
   7.62828940163412l9E+004   4.90619611874225l8E+004   1.54298485382269l9E+004
   9.64824086468674l2E+002
  1.00000000
  0.00000000
C ********** Beta Mode **********
C 1202 - (1202L), 3 bundle 1590 SDC TYPE 7/9
-1B____2B____1                2.  0.00              -2 1
        6       2.28629697636174260lE+002
   1.09308752661520l8E+004   3.78169774519262l3E+002   7.22447424546670l0E+002
   4.19507826764704l9E+002   2.25553354800l5390E+002   1.56902118589791l0E+002
   6.17382618753010l2E+003   2.42721391328913l8E+001   4.27309979785709l5E+000
   1.10909963645687l4E+000   2.84286299431262l8E-001   1.02289435430268l5E-001
        7       2.30795136251357390lE-004
   1.11054222231736l0E+006   3.83833268403616l8E+004   1.33903607258887l0E+003
   3.23199946329516l2E+002   2.51189159935285l1E+002   7.70289318758678l5E+001
   4.47011148418639l7E+000
   1.23991319767138l0E+006   5.82329720729196l9E+005   6.82087978589703l6E+004
   7.77558737106478l2E+004   5.00093719936714l1E+004   1.63431060998878l0E+004
   9.64763965874237l8E+002
  1.00000000
  0.00000000
C 1203 - (1203L), 3 bundle DRAKE
-1B____2NN524B                2.  0.00              -2 1
        6       2.35229445783656l000E+002
   3.95059139542088889E+004   8.01974122254ll199E+002   1.47251423773319l6E+003
   8.22016760521079l6E+002   4.44135227292783l8E+002   2.77941793750997l6E+002
   2.66809360982867l9E+004   4.49211411400484473E+001   7.78322785913622l1E+000
   1.86937975998527l1E+000   4.27694837032311l7E-001   1.11872929265317l7E-001
        5       6.10422368881328920lE-005
   2.41516096734507l0E+006  -2.82306923430l8622E+004   7.44494515752063l8E+002
   7.27454626704187l8E+002   6.14644360804775l2E+001
   2.44114964668l7860E+006   4.57971485248l8365E+006   1.38918887734267l1E+005
   1.23176856272740l0E+005   1.12879665693264l1E+004
  1.00000000
  0.00000000
C 1209 - (1209L), 3 bundle 1590 SDC TYPE 7/9, TRILLIUM
-1B____1NN524B                2.  0.00              -2 1
        6       2.28629697636174260lE+002
   1.09308752661520l8E+004   3.78169774519262l3E+002   7.22447424546670l0E+002
   4.19507826764704l9E+002   2.25553354800l5390E+002   1.56902118589791l0E+002
   6.17382618753010l2E+003   2.42721391328913l8E+001   4.27309979785709l5E+000
   1.10909963645687l4E+000   2.84286299431262l8E-001   1.02289435430268l5E-001
        7       2.24555200482619610lE-004
   9.98874107240689l0E+006   1.10086473725561l0E+005   1.35200927653972l0E+003
   2.89287468770128l8E+002   2.38000457034562l1E+002   7.06678707780972l0E+001
   4.35059666741596l3E+000
   1.19442152421977l0E+006   8.71736957821651l3E+005   6.95475731557374l7E+004
   7.62828940163412l9E+004   4.90619611874225l8E+004   1.54298485382269l9E+004
   9.64824086468674l2E+002
  1.00000000
  0.00000000
C ************
C deep region
C ************
C BEGIN FORE
C ********** Alpha Mode **********
$VINTAGE,1
C <BUS1><BUS2><BUS3><BUS4><  OHM        ><   milliH     ><   microF     >
C
C (1,1)
  A____1                   4.794415e+001
  A____1                   9.493358e+000   1.713512e+002
  A____1                   2.749832e+003   2.027777e+004
  A____1                  -4.959225e+001  -3.649759e+000
  A____1A 4__1             -7.177667e+001  -7.209291e+002
  A 4__1                    1.144283e+002
  A 4__1                                                  -3.579021e+002
  A____1A 6__1              4.891120e+000   1.096120e+002
  A 6__1                   -1.460729e+005
  A 6__1                                                   7.690014e+000
  A____1A 8__1              5.273031e+000   5.655519e+001
  A 8__1                   -6.603677e+003
  A 8__1                                                   4.595994e+000
  A____1A A__1             -8.781766e-001   1.404059e+002
  A A__1                    2.389376e+004
  A A__1                                                   6.859620e-001
  A____1A C__1              1.605895e+002   3.123332e+002
  A C__1                   -9.180542e+003
  A C__1                                                   2.509075e-001
  A____1A E__1              5.796004e+000   5.743626e+001
  A E__1                   -4.518826e+004
  A E__1                                                   1.123840e+000
  A____1A10__1              5.347305e+000   7.876362e+001
  A10__1                    7.120424e+004
  A10__1                                                  6.714215e-001
  A____1A12__1             -4.031501e+002   1.320121e+003
  A12__1                    8.496503e+004
  A12__1                                                  2.756715e-002
  A____1A14__1              9.837555e+000   4.307456e+001
  A14__1                   -1.224322e+004
  A14__1                                                  6.348458e-001
  A____1A16__1              2.723430e+001   2.055907e+001
  A16__1                   -2.016991e+003
  A16__1                                                  4.175572e-001
  A____1A18__1              9.151733e+000   3.943101e+001
  A18__1                   -7.115812e+004
  A18__1                                                  1.373400e-001
C (1,2)
  A____1A____2             -2.412135e+004
  A____1A____2              1.264011e+001   2.281487e+002
  A____1A____2              1.319321e+002   9.728918e+002
  A____1A____2             -8.499067e+003  -6.254918e+002
  A____1A 4_12              2.894614e+001   5.290562e+002
  A 4_12A____2              8.461279e+000
  A 4_12A____2                                            5.784619e+003
  A____1A 6_12             -1.400855e+001  -3.397235e+002
  A 6_12A____2             -1.614046e+005
  A 6_12A____2                                           -2.481487e+000
  A____1A 8_12             -1.649082e+001  -1.749592e+002
  A 8_12A____2              1.981734e+004
  A 8_12A____2                                           -1.485595e+000
  A____1A A_12              4.669506e+000  -4.794408e+002
  A A_12A____2             -7.718285e+004
  A A_12A____2                                           -2.008815e-001
  A____1A C_12             -3.786232e+002  -9.427005e+002
  A C_12A____2              3.712853e+004
  A C_12A____2                                           -8.374724e-002
  A____1A E_12              1.583677e+002   1.118122e+003
```

```
A E_12A____2              -2.868322e+005
A E_12A____2                                          5.770550e-002
A____1A10_12               2.495367e+001   2.351903e+002
A10_12A____2              -2.572280e+005
A10_12A____2                                          2.248158e-001
A____1A12_12               5.012797e+002  -2.021988e+003
A12_12A____2              -1.501703e+005
A12_12A____2                                         -1.802356e-002
A____1A14_12              -2.394826e+001  -1.030946e+002
A14_12A____2               2.843980e+004
A14_12A____2                                         -2.652384e-001
A____1A16_12              -3.029497e+003   4.894610e+002
A16_12A____2               1.191969e+004
A16_12A____2                                          1.326025e-002
A____1A18_12              -2.675602e+002   3.302770e+002
A18_12A____2               6.514827e+004
A18_12A____2                                          1.633147e-002
C (2,2)
A____2                     1.561783e+002
A____2                     3.399620e+001   6.136174e+002
A____2                    -1.611083e+003  -1.188042e+004
A____2                    -1.649564e+002  -1.214002e+001
A____2A 4__2              -1.284818e+001   1.715270e+003
A 4__2                     4.283411e+001
A 4__2                                                2.825206e+002
A____2A 6__2               9.128618e+000   2.302783e+002
A 6__2                     6.678028e+004
A 6__2                                                3.661054e+000
A____2A 8__2               1.133786e+001   1.184714e+002
A 8__2                    -1.287173e+004
A 8__2                                                2.193829e+000
A____2A A__2              -1.158493e+000   3.389292e+002
A A__2                     6.048886e+004
A A__2                                                2.841737e-001
A____2A C__2               2.355020e+002   6.303375e+002
A C__2                    -2.715400e+004
A C__2                                                1.254410e-001
A____2A E__2              -1.029387e+002  -1.176880e+003
A E__2                     2.918482e+006
A E__2                                               -5.485280e-002
A____2A10__2              -2.393039e+001  -3.143175e+002
A10__2                    -4.690046e+005
A10__2                                               -1.682449e-001
A____2A12__2               1.814293e+002   6.146666e+002
A12__2                    -9.700521e+004
A12__2                                                5.937703e-002
A____2A14__2               1.484779e+001   6.005530e+001
A14__2                    -1.489116e+004
A14__2                                                4.552538e-001
A____2A16__2               4.956775e+003  -5.222964e+002
A16__2                    -1.119085e+004
A16__2                                               -9.281440e-003
A____2A18__2               2.830843e+002  -3.804930e+002
A18__2                    -8.068415e+004
A18__2                                               -1.418462e-002
$VINTAGE,0
C ********************************** Beta Mode ********************************
$VINTAGE,1
C <BUS1><BUS2><BUS3><BUS4><   OHM        ><   milliH     ><   microF     >
C
C (1,1)
B____1                     4.794415e+001
B____1                     9.493358e+000   1.713512e+002
B____1                     2.749832e+003   2.027777e+004
B____1                    -4.959225e+001  -3.649759e+000
B____1B 4__1              -7.177667e+001  -7.209291e+002
B 4__1                     1.144283e+002
B 4__1                                               -3.579021e+002
B____1B 6__1               4.891120e+000   1.096120e+002
B 6__1                    -1.460729e+005
B 6__1                                                7.690014e+000
B____1B 8__1               5.273031e+000   5.955519e+001
B 8__1                    -6.603677e+003
B 8__1                                                4.595994e+000
B____1B A__1              -8.781766e-001   1.404059e+002
B A__1                     2.389376e+004
B A__1                                                6.859620e-001
B____1B C__1               1.605895e+002   3.123332e+002
B C__1                    -9.180542e+003
B C__1                                                2.509075e-001
B____1B E__1               5.796004e+000   5.743626e+001
B E__1                    -4.518826e+004
B E__1                                                1.123840e+000
B____1B10__1               5.347305e+000   7.876362e+001
B10__1                     7.120424e+004
B10__1                                                6.714215e-001
B____1B12__1              -4.031501e+002   1.320121e+003
B12__1                     8.496503e+004
B12__1                                                2.756715e-002
B____1B14__1               9.837555e+000   4.307456e+001
B14__1                    -1.224322e+004
B14__1                                                6.348458e-001
B____1B16__1               2.723430e+001   2.055907e+001
B16__1                    -2.016991e+003
B16__1                                                4.175572e-001
B____1B18__1               9.151733e+000   3.943101e+001
B18__1                    -7.115812e+004
B18__1                                                1.373400e-001
C (1,2)
B____1B____2              -2.412135e+004
B____1B____2               1.264011e+001   2.281487e+002
B____1B____2               1.319321e+002   9.728918e+002
B____1B____2              -8.499067e+003  -6.254918e+002
B____1B 4_12              2.894614e+001   5.290562e+002
B 4_12B____2              8.461279e+000
B 4_12B____2                                          5.784519e-003
B____1B 6_12              -1.400855e+001  -3.397235e+002
B 6_12B____2              -1.614046e+005
B 6_12B____2                                         -2.481487e+000
B____1B 8_12              -1.649082e+001  -1.749592e+002
B 8_12B____2               1.981734e+004
B 8_12B____2                                         -1.485595e+000
B____1B A_12               4.669506e+000  -4.794408e+002
B A_12B____2              -7.718285e+004
B A_12B____2                                         -2.008815e-001
B____1B C_12              -3.786232e+002  -9.427005e+002
B C_12B____2               3.712853e+004
B C_12B____2                                         -8.374724e-002
B____1B E_12               1.583677e+002   1.118122e+003
B E_12B____2              -2.868322e+005
B E_12B____2                                          5.770550e-002
B____1B10_12               2.495367e+001   2.351903e+002
B10_12B____2              -2.572280e+005
B10_12B____2                                          2.248158e-001
B____1B12_12               5.012797e+002  -2.021988e+003
B12_12B____2              -1.501703e+005
B12_12B____2                                         -1.802356e-002
B____1B14_12              -2.394826e+001  -1.030946e+002
B14_12B____2               2.843980e+004
B14_12B____2                                         -2.652384e-001
B____1B16_12              -3.029497e+003   4.894610e+002
B16_12B____2               1.191969e+004
B16_12B____2                                          1.326025e-002
B____1B18_12              -2.675602e+002   3.302770e+002
B18_12B____2               6.514827e+004
B18_12B____2                                          1.633147e-002
C (2,2)
B____2                     1.561783e+002
B____2                     3.399620e+001   6.136174e+002
B____2                    -1.611083e+003  -1.188042e+004
B____2                    -1.649564e+002  -1.214002e+001
B____2B 4__2              -1.284818e+001   1.715270e+003
B 4__2                     4.283411e+001
B 4__2                                                2.825206e+002
B____2B 6__2               9.128618e+000   2.302783e+002
B 6__2                     6.678028e+004
```

```
B 6__2
B__2B 8__2
B 8__2
B__2B A__2
B A__2
B__2B C__2
B C__2
B__2B E__2
B E__2
B__2B10__2
B10__2
B__2B12__2
B12__2
B__2B14__2
B14__2
B__2B16__2
B16__2
B__2B18__2
B18__2
$VINTAGE,0
C END FDNE
/SWITCH
```

```
 1.133786e+001  1.184714e+002  3.661054e+000
-1.267173e+004
-1.158493e+000  3.389292e+002  2.193825e+000
 6.048886e+004
 2.355020e+002  6.303375e+002  2.841737e-001
-2.715400e+004
-1.029387e+002 -1.176880e+003  1.254410e-001
 2.918402e+006
-2.393039e+001 -3.143175e+002 -5.485280e-002
-4.690046e+005
-1.814293e+002  6.146666e+002 -1.682449e-001
-9.700521e+004
-1.484779e+001  6.005536e+001  5.937703e-002
-1.489116e+004
 4.956775e+003 -5.222964e+002  4.552538e-001
-1.119085e+004
 2.830843e+002 -3.804930e+002 -9.281440e-003
-8.068415e+004
                               -1.418462e-002
```

```
C  < n 1><  n 2>< Iclose ><Top/Tde ><   Re   ><VR/CLOP ><  type >
C  ************************************ Alpha Mode ********************************
C    MN524AX0030A          .05          1.
C  ************************************ Beta Mode *********************************
C    MN524BX0030B          .05          1.
/SOURCE
C  < n 1><  Ampl.  >< Freq.  ><Phase/TD>< A1  >< T1  >< TSTART >< TSTOP >
C  ************************************ Alpha Mode ********************************
 14X0018A 0     254.38        60.          20.5                    -1.          1.
 14X0020A 0     254.38        60.          29.9                    -1.          1.
C  ************************************
C  Norton equivalent current source
C  ************************************
C  14MN524A-12.30377594          60. -125.6348                      -1.
C  ************************************ Beta Mode *********************************
 14X0018B 0     254.38        60.         -61.5                    -1.          1.
 14X0020B 0     254.38        60.         -60.1                    -1.          1.
C  ************************************
C  Norton equivalent current source
C  ************************************
C  14MN524B-12.30377594          60. -215.6348                      -1.          1.
/OUTPUT
/SUBCKT MN524BANN524B
BLANK BRANCH
BLANK SWITCH
BLANK SOURCE
BLANK OUTPUT
BLANK PLOT
BEGIN NEW DATA CASE
BLANK
```

D

EMTP Models of Passive Elements

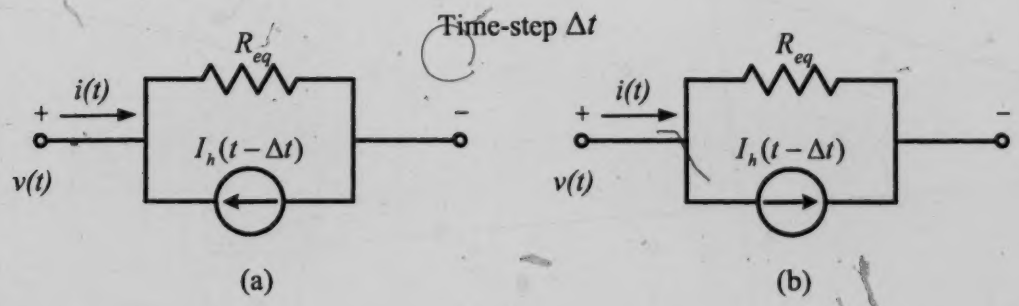


Figure D.1: Norton equivalent circuit of the discretized branch of passive elements

D.1 L Branch

Shown in Fig. D.1(b)

$$i(t) = \frac{1}{R_{eq}} v(t) - I(t - \Delta t)$$

where

$$R_{eq} = \frac{2L}{\Delta t}$$

$$I_h(t - \Delta t) = I_h(t - 2\Delta t) + \frac{2}{R_{eq}} v(t - \Delta t)$$

D.2 C Branch

Shown in Fig. D.1(a)

$$i(t) = \frac{1}{R_{eq}} v(t) + I(t - \Delta t)$$

where

$$R_{eq} = \frac{\Delta t}{2C}$$

$$I_h(t - \Delta t) = -I_h(t - 2\Delta t) + \frac{2}{R_{eq}} v(t - \Delta t)$$

D.3 RL Branch

Shown in Fig. D.1(b)

$$i(t) = \frac{1}{R_{eq}} v(t) - I(t - \Delta t)$$

where

$$R_{eq} = R + \frac{2L}{\Delta t}$$

$$I_h(t - \Delta t) = -\frac{R - \frac{2L}{\Delta t}}{R_{eq}} I_h(t - 2\Delta t) + \frac{\frac{4L}{\Delta t}}{R_{eq}^2} v(t - \Delta t)$$

D.4 RC Branch

Shown in Fig. D.1(a)

$$i(t) = \frac{1}{R_{eq}} v(t) + I(t - \Delta t)$$

where

$$R_{eq} = R + \frac{\Delta t}{2C}$$

$$I_h(t - \Delta t) = -\frac{R - \frac{\Delta t}{2C}}{R_{eq}} I_h(t - 2\Delta t) + \frac{2R}{R_{eq}^2} v(t - \Delta t)$$

D.5 LC Branch

Shown in Fig. D.1(a)

$$i(t) = \frac{1}{R_{eq}} v(t) + I(t - \Delta t)$$

where

$$R_{eq} = \frac{2L}{\Delta t} + \frac{\Delta t}{2C}$$

$$I_h(t-\Delta t) = \frac{V_{h,L}(t-\Delta t) + V_{h,C}(t-\Delta t)}{R_{eq}}$$

$$V_{h,L}(t-\Delta t) = -V_{h,L}(t-2\Delta t) - \frac{4L}{\Delta t} i(t-\Delta t)$$

$$V_{h,C}(t-\Delta t) = V_{h,C}(t-2\Delta t) + \frac{\Delta t}{C} i(t-\Delta t)$$

D.6 *RLC* Branch

Shown in Fig. D.1(a)

$$i(t) = \frac{1}{R_{eq}} v(t) + I(t-\Delta t)$$

where

$$R_{eq} = R + \frac{2L}{\Delta t} + \frac{\Delta t}{2C}$$

$$I_h(t-\Delta t) = \frac{V_{h,L}(t-\Delta t) + V_{h,C}(t-\Delta t)}{R_{eq}}$$

$$V_{h,L}(t-\Delta t) = -V_{h,L}(t-2\Delta t) - \frac{4L}{\Delta t} i(t-\Delta t)$$

$$V_{h,C}(t-\Delta t) = V_{h,C}(t-2\Delta t) + \frac{\Delta t}{C} i(t-\Delta t)$$

D.7 *RLCG* Branch

Shown in Fig. D.1(a)

$$i(t) = \frac{1}{R_{eq}} v(t) + I(t-\Delta t)$$

where

$$R_{eq} = R + \frac{2L}{\Delta t} + \frac{G\frac{\Delta t}{2C}}{G + \frac{\Delta t}{2C}}$$

$$I_h(t-\Delta t) = \frac{V_{h,L}(t-\Delta t) + V_{h,CG}(t-\Delta t)}{R_{eq}}$$

$$V_{h,L}(t-\Delta t) = -V_{h,L}(t-2\Delta t) - \frac{4L}{\Delta t} i(t-\Delta t)$$

$$V_{h,CG}(t-\Delta t) = \frac{G - \frac{\Delta t}{2C}}{G + \frac{\Delta t}{2C}} V_{h,CG}(t-2\Delta t) + \frac{G^2 \frac{\Delta t}{C}}{(G + \frac{\Delta t}{2C})^2} i(t-\Delta t)$$

Note that G is in Ω.

E

C++ EMTP S-function Complete Source Code

The S-function program includes two files:

- `emtp.h`. This is the header file that defines all class and structures, and implements all functions including reading ATP data files and auxiliary functions.
- `emtp.cpp`. This is the main S-function program that implements S-function initialization, simulation loop and output, as well as EMTP initialization, simulation loop and solutions.

E.1 emtp.h

```
/*  File    : emtp.h
 *  Abstract:
 *
 *      Header file for C++ implementation of SimuLINK S-function
 *      for Real-time EMTP
 *
 *  by Xin Nie, Power Engineering Group
 *  Dept. of Electrical and Computer Engineering
 *  University of Alberta, Edmonton, Canada
 *  email: xnie@ece.ualberta.ca
 *  May 10, 2005
 */

#ifndef EMTP_H
#define EMTP_H

#include <math.h>
#include "matrix.h"

typedef math::matrix<double> CMatrix;

static const double PI = 3.1415927;
static const double Rinfs = 1.0e-10;        // infinite small resistance
static const double Rinfl = 1.0e10;         // infinite large resistance

// the abstract base class for other circuit element to derive
// this is an abstract class and cannot be instantiated

// Passive circuit element
class CPElement {
public:
    virtual void update(double&) = 0;       // update history terms
    virtual double getG() = 0;              // get conductance G
    virtual double getIh() = 0;             // get history term Ih
    virtual double getib() = 0;             // get branch current ib
};

// Active circuit element
class CAElement {
public:
    virtual double getG() = 0;              // get conductance G
    virtual double getIeq(double&) = 0;     // get history term I
    virtual double getib(double&, double&) = 0; // get branch current
};

// Transmission line element
class CTLElement {
public:
    virtual double getIhk() = 0;            // get history term Ihk
    virtual double getIhm() = 0;            // get history term Ihm
    virtual void update(double&, double&) = 0;  // update history terms
    virtual double getG() = 0;              // get conductance G
    virtual double getik() = 0;             // get branch current ik
    virtual double getim() = 0;             // get branch current im
};

// Switch element
class CSWElement {
public:
    virtual double getG() = 0;              // get conductance G
    virtual double getib(double&) = 0;      // get branch current
    virtual void update(bool&) = 0;         // update history terms
};

// get switch conductance G
/*
    st defines switch states
    0/false --- close
    1/true  --- open
*/

double getSWG(bool &st) {
    return st?(1.0/Rinfl):(1.0/Rinfs);
}

/*************************/
/*    Switch Elements    */
/*************************/

// class for switchs
class CSwitch : public CSWElement {
    bool st;
public:
    double getG();
    double getib(double &v);
    void update(bool &st);
};

inline void CSwitch::update(bool &sts) {
    st = sts;
}

inline double CSwitch::getib(double &v) {
    return v*getG();
}

inline double CSwitch::getG() {
    return getSWG(st);
}

/*************************/
/*    Passive Elements   */
/*************************/

// class for R branch
// o----R----o
class CR : public CPElement {
    double R;
    double ib;
public:
    CR(double &Rs);
    void update(double &v);
    double getG();
    double getIh();
    double getib();
};

inline CR::CR(double &Rs) {
    R = Rs;
}

inline void CR::update(double &v) {
    ib = v/R;
}

inline double CR::getG() {
    return 1.0/R;
}

inline double CR::getIh() {
    return 0.0;
}

inline double CR::getib() {
    return ib;
}

// class for L branch
// o----L----o
class CL : public CPElement {
    double A, B;
    double Req;
    double Ih;
    double ib;
```

```
144 public:
145     CL(double &L, double &dt);
146     void update(double &v);
147     double getG();
148     double getIh();
149     double getib();
150 };
151
152 inline CL::CL(double &L, double &dt) {
153     // discretization
154     Req = 2.0*L/dt;
155     // history term
156     // I(t-dt) = A*I(t-2*dt)+B*v(t-dt)
157     A = 1;
158     B = 2.0/Req;
159     Ih = 0.0;
160 }
161
162 inline void CL::update(double &v) {
163     ib = v/Req + Ih;
164     Ih = A * Ih + B * v;
165 }
166
167 inline double CL::getG() {
168     return 1.0/Req;
169 }
170
171 inline double CL::getIh() {
172     return -Ih;
173 }
174
175 inline double CL::getib() {
176     return ib;
177 }
178
179 // class for C branch
180 // o----C----o
181 class CC : public CPElement {
182     double A, B;
183     double Req;
184     double Ih;
185     double ib;
186 public:
187     CC(double &C, double &dt);
188     void update(double &v);
189     double getG();
190     double getIh();
191     double getib();
192 };
193
194 inline CC::CC(double &C, double &dt) {
195     // discretization
196     Req = dt/(2.0*C);
197     // history term
198     // I(t-dt) = A*I(t-2*dt)+B*v(t-dt)
199     A = -1;
200     B = 2.0/Req;
201     Ih = 0.0;
202 }
203
204 inline void CC::update(double &v) {
205     ib = v/Req - Ih;
206     Ih = A * Ih + B * v;
207 }
208
209 inline double CC::getG() {
210     return 1.0/Req;
211 }
212
213 inline double CC::getIh() {
214     return Ih;
215 }
216
217 inline double CC::getib() {
218     return ib;
219 }
220
221 // class for RL branch
222 // o----R--L----o
223 class CRL : public CPElement {
224     double A, B;
225     double Req;
226     double Ih;
227     double ib;
228 public:
229     CRL(double &R, double &L, double &dt);
230     void update(double &v);
231     double getG();
232     double getIh();
233     double getib();
234 };
235
236 inline CRL::CRL(double &R, double &L, double &dt) {
237     // discretization
238     double Ld = 2.0*L/dt;
239     Req = R+Ld;
240     // history term
241     // I(t-dt) = A*I(t-2*dt)+B*v(t-dt)
242     A = -(R-Ld)/Req;
243     B = (2.0*Ld)/(Req*Req);
244     Ih = 0.0;
245 }
246
247 inline void CRL::update(double &v) {
248     ib = v/Req + Ih;
249     Ih = A * Ih + B * v;
250 }
251
252 inline double CRL::getG() {
253     return 1.0/Req;
254 }
255
256 inline double CRL::getIh() {
257     return -Ih;
258 }
259
260 inline double CRL::getib() {
261     return ib;
262 }
263
264 // class for RC branch
265 // o----R--C----o
266 class CRC : public CPElement {
267     double A, B;
268     double Req;
269     double Ih;
270     double ib;
271 public:
272     CRC(double &R, double &C, double &dt);
273     void update(double &v);
274     double getG();
275     double getIh();
276     double getib();
277 };
278
279 inline CRC::CRC(double &R, double &C, double &dt) {
280     // discretization
281     double Cd = dt/(2.0*C);
282     Req = R+Cd;
283     // history term
284     // I(t-dt) = A*I(t-2*dt)+B*v(t-dt)
285     A = -(R-Cd)/Req;
286     B = (2.0*R)/(Req*Req);
287     Ih = 0.0;
288 }
289
```

```
290 inlihe void CRC::update(double &v) {
291     ib = v/Req - Ih;
292     Ih = A * Ih + B * v;
293 }
294
295 inline double CRC::getG() {
296     return 1.0/Req;
297 }
298
299 inline double CRC::getIh() {
300     return Ih;
301 }
302
303 inline double CRC::getib() {
304     return ib;
305 }
306
307 // class for LC branch
308 // o----L--C----o
309 class CLC : public CPElement {
310     double A[2], B[2], Vh[2];
311     double Req,ib;
312 public:
313     CLC(double &L, double &C, double &dt);
314     void update(double &v);
315     double getIh();
316     double getG();
317     double getib();
318 };
319
320 inline CLC::CLC(double &L, double &C, double &dt) {
321     // discretization
322     double Ld = 2.0*L/dt;
323     double Cd = dt/(2.0*C);
324     Req = Ld+Cd;
325     // history term
326     // I(t-dt) = A*I(t-2*dt)+B*v(t-dt)
327     A[0] = -1.0;
328     B[0] = -2.0*Ld;
329     Vh[0] = 0.0;
330     A[1] = 1;
331     B[1] = 2.0*Cd;
332     Vh[1] = 0.0;
333 }
334
335 inline void CLC::update(double &v) {
336     ib = v/Req - getIh();
337     Vh[0] = A[0]*Vh[0] + B[0]*ib;
338     Vh[1] = A[1]*Vh[1] + B[1]*ib;
339 }
340
341 inline double CLC::getIh() {
342     return (Vh[0]+Vh[1])/Req;
343 }
344
345 inline double CLC::getG() {
346     return 1.0/Req;
347 }
348
349 inline double CLC::getib() {
350     return ib;
351 }
352
353 // class for RLC branch
354 // o----R--L--C----o
355 class CRLC : public CPElement {
356     double A[2], B[2], Vh[2];
357     double Req,ib;
358 public:
359     CRLC(double &R, double &L, double &C, double &dt);
360     void update(double &v);
361     double getIh();
362     double getG();
363     double getib();
364 };
365
366 inline CRLC::CRLC(double &R, double &L, double &C, double &dt) {
367     // discretization
368     double Ld = 2.0*L/dt;
369     double Cd = dt/(2.0*C);
370     Req = R+Ld+Cd;
371     // history term
372     // I(t-dt) = A*I(t-2*dt)+B*v(t-dt)
373     A[0] = -1.0;
374     B[0] = -2.0*Ld;
375     Vh[0] = 0.0;
376     A[1] = 1;
377     B[1] = 2.0*Cd;
378     Vh[1] = 0.0;
379 }
380
381 inline void CRLC::update(double &v) {
382     ib = v/Req - getIh();
383     Vh[0] = A[0]*Vh[0] + B[0]*ib;
384     Vh[1] = A[1]*Vh[1] + B[1]*ib;
385 }
386
387 inline double CRLC::getIh() {
388     return (Vh[0]+Vh[1])/Req;
389 }
390
391 inline double CRLC::getG() {
392     return 1.0/Req;
393 }
394
395 inline double CRLC::getib() {
396     return ib;
397 }
398
399 // RLCG branch
400 //                ---C---
401 // o----R--L--|     |----o
402 //                ---G---
403 class CRLCG : public CPElement {
404     double A[2], B[2], Vh[2];
405     double Req,ib;
406 public:
407     CRLCG(double &R, double &L, double &C, double &G, double &dt);
408     void update(double &v);
409     double getIh();
410     double getG();
411     double getib();
412 };
413
414 inline CRLCG::CRLCG(double &R, double &L, double &C, double &G, double &dt) {
415     // discretization
416     double Ld = 2.0*L/dt;
417     double Cd = dt/(2.0*C);
418     Req = R+Ld+G*Cd/(G+Cd);
419     // history term
420     // V(t-dt) = A*V(t-2*dt)+B*i(t-dt)
421     A[0] = -1.0;
422     B[0] = -2.0*Ld;
423     Vh[0] = 0.0;
424     A[1] = (G-Cd)/(G+Cd);
425     B[1] = 2.0*G*G*Cd/((G+Cd)*(G+Cd));
426     Vh[1] = 0.0;
427 }
428
429 inline void CRLCG::update(double &v) {
430     ib = v/Req - getIh();
431     Vh[0] = A[0]*Vh[0] + B[0]*ib;
432     Vh[1] = A[1]*Vh[1] + B[1]*ib;
433 }
434
435 inline double CRLCG::getIh() {
```

```
436     return (Vh[0]+Vh[1])/Req;
437 }
438
439 inline double CRLCG::getG() {
440     return 1.0/Req;
441 }
442
443 inline double CRLCG::getib() {
444     return ib;
445 }
446
447 /*************************/
448 /*  Trans. Line Elements  */
449 /*************************/
450
451 // Marti's frequency-dependent transmission line model
452 class CFdt1 : public CTLElement {
453     double *Vk, *Vm, *Fk, *Fm, *Sk, *Sm;
454     double *A, *B, *M, *P, *Q, *K;
455     double Vks, Vms, Bk, Bm;
456     double dt;
457     const int NZc, NP, NTau;
458     void ArrangeF();
459     double Zceq;
460     double ik, im;
461 public:
462     CFdt1(double &ZcCon, double* ZcRes, double* ZcPol, int sNZc,
463         double &Tau, double* PRes, double* PPol, int sNP, double &dt);
464     double getIhk();
465     double getIhm();
466     void update(double &vk, double &vm);
467     void updateBkm();
468     double getG();
469     double getik();
470     double getim();
471     ~CFdt1();
472 };
473
474 CFdt1::CFdt1(double &ZcCon,                 // constant term of Zc
475              double* ZcRes,                 // residues of Zc
476              double* ZcPol,                 // poles of Zc
477              int sNZc,                      // order of Zc
478              double &Tau,                   // traveling time
479              double* PRes,                  // residues of A1/P
480              double* PPol,                  // poles of A1/P
481              int sNP,                       // order of Zc
482              double &sdt) : NZc(sNZc), NP(sNP), NTau((int) (Tau/sdt)) {
483     double R, C, Cd, alp, p, Zt;
484     int i;
485     dt = sdt;
486     A = new double[NZc];
487     B = new double[NZc];
488     M = new double[NP];
489     P = new double[NP];
490     Q = new double[NP];
491     K = new double[NP];
492     // history terms
493     Vk = new double[NZc];
494     Vm = new double[NZc];
495     Fk = new double[NTau+1];
496     Fm = new double[NTau+1];
497     Sk = new double[NP];
498     Sm = new double[NP];
499
500     // discretized Characteristic Impedance coefficients
501     Zt = ZcCon;
502     for(i=0;i<NZc;i++) {
503         // find equivalent circuit
504         R = ZcRes[i]/ZcPol[i];
505         C = 1/ZcRes[i];
506         // discretization
507         Cd = dt/(2.0*C);
508         Zt += R*Cd/(R+Cd);
509         // V(t-dt) = A*V(t-2*dt)+B*i(t-dt)
510         A[i] = (R-Cd)/(R+Cd);
511         B[i] = 2.0*R*R*Cd/((R+Cd)*(R+Cd));
512         Vk[i] = 0.0;
513         Vm[i] = 0.0;
514     }
515     Zceq=Zt;
516     // Recursive Convolution coefficients
517     for(i=0;i<NP;i++) {
518         // b(t)=M*b(t-dt)+P*f(t-tau)+Q*f(t-tau-dt)
519         p=PPol[i];
520         alp = exp(-p*dt);
521         M[i] = alp;
522         P[i] = 1/p*(1-(1-alp)/(p*dt));
523         Q[i] = 1/p*(((1-alp)/(p*dt))-alp);
524         Sk[i] = 0.0;
525         Sm[i] = 0.0;
526         K[i] = PRes[i];
527     }
528     for(i=0;i<NTau+1;i++) {
529         Fk[i] = 0.0;
530         Fm[i] = 0.0;
531     }
532     Vks = 0.0;
533     Vms = 0.0;
534     Bk = 0.0;
535     Bm = 0.0;
536 }
537
538 // arrange the two Forward traveling function so that the first
539 // element of the array is always Fk(t-tau-dt) and Fm(t-tau-dt)
540 inline void CFdt1::ArrangeF() {
541     double Fkt = Fk[0];
542     double Fmt = Fm[0];
543     for(int i=0;i<NTau;i++) {
544         Fk[i] = Fk[i+1];
545         Fm[i] = Fm[i+1];
546     }
547     Fk[NTau] = Fkt;
548     Fm[NTau] = Fmt;
549 }
550
551 inline double CFdt1::getIhk() {
552     return (Vks+Bk)/Zceq;
553 }
554
555 inline double CFdt1::getIhm() {
556     return (Vms+Bm)/Zceq;
557 }
558
559 inline double CFdt1::getG() {
560     return 1.0/Zceq;
561 }
562
563 inline double CFdt1::getik() {
564     return ik;
565 }
566
567 inline double CFdt1::getim() {
568     return im;
569 }
570
571 inline void CFdt1::update(double &vk, double &vm) {
572     ik = vk/Zceq-getIhk();
573     im = vm/Zceq-getIhm();
574     Vks = 0.0;
575     Vms = 0.0;
576     // update history terms of Zc
577     for(int i=0;i<NZc;i++) {
578         Vk[i] = A[i] * Vk[i] + B[i] * ik;
579         Vm[i] = A[i] * Vm[i] + B[i] * im;
580         Vks += Vk[i];
581         Vms += Vm[i];
```

```
582     }
583     Fk[0] = 2.0 * vk - Bk;
584     Fm[0] = 2.0 * vm - Bm;
585     ArrangeF();
586 }
587
588 // recursive convolution
589 inline void CFdtl::updateBkm() {
590     Bk = 0.0;
591     Bm = 0.0;
592     for(int i=0;i<NP;i++) {
593         Sk[i] = M[i] * Sk[i] + P[i] * Fm[i] + Q[i] * Fm[0];
594         Bk += K[i] * Sk[i];
595         Sm[i] = M[i] * Sm[i] + P[i] * Fk[i] + Q[i] * Fk[0];
596         Bm += K[i] * Sm[i];
597     }
598 }
599
600 inline CFdtl::~CFdtl() {
601     delete []A;
602     delete []B;
603     delete []M;
604     delete []P;
605     delete []Q;
606     delete []K;
607     delete []Vk;
608     delete []Vm;
609     delete []Fk;
610     delete []Fm;
611     delete []Sk;
612     delete []Sm;
613 }
614
615 /*************************/
616 /*    Active Elements    */
617 /*************************/
618
619 // voltage source with resistance
620 class CVsr : public CAElement {
621     const double mag, f, pha;
622     const double R;
623 public:
624     CVsr(double &Rs, double &mags, double &fs, double &phas);
625     double getIeq(double &t);
626     double getG();
627     double getib(double &v, double &t);
628 };
629
630 CVsr::CVsr(double &Rs, double &mags, double &fs,
631            double &phas) : R(Rs), mag(mags), f(fs), pha(phas) {}
632
633 inline double CVsr::getIeq(double &t) {
634     return (mag*cos(2.0*PI*f*t+pha*PI/180))/R;
635 }
636
637 inline double CVsr::getG() {
638     return 1.0/R;
639 }
640
641 inline double CVsr::getib(double &v, double &t) {
642     return (getIeq(t)-v/R);
643 }
644
645 // ideal voltage source
646 class CVs : public CAElement {
647     const double mag, f, pha;
648     const double R;
649 public:
650     CVs(double &mags, double &fs, double &phas);
651     double getIeq(double &t);
652     double getG();
653     double getib(double &v, double &t);
654 };
655
656 CVs::CVs(double &mags, double &fs,
657          double &phas) : R(Rinfs), mag(mags), f(fs), pha(phas) {}
658
659 inline double CVs::getIeq(double &t) {
660     return (mag*cos(2.0*PI*f*t+pha*PI/180))/R;
661 }
662
663 inline double CVs::getG() {
664     return 1.0/R;
665 }
666
667 inline double CVs::getib(double &v, double &t) {
668     return (getIeq(t)-v/R);
669 }
670
671 // ideal current source
672 class CIs : public CAElement {
673     const double mag, f, pha;              // phase shift in degree
674     const double R;
675 public:
676     CIs(double &mags, double &fs, double &phas);
677     double getIeq(double &t);
678     double getG();
679     double getib(double &v, double &t);
680 };
681
682 CIs::CIs(double &mags, double &fs,
683          double &phas) : R(Rinfl), mag(mags), f(fs), pha(phas) {}
684
685 inline double CIs::getIeq(double &t) {
686     return mag*cos(2.0*PI*f*t+pha*PI/180);
687 }
688
689 inline double CIs::getG() {
690     return 0.0;
691 }
692
693 inline double CIs::getib(double &v, double &t) {
694     return getIeq(t);
695 }
696
697 /*************************/
698 /*  Auxiliary functions  */
699 /*************************/
700
701 // double to bool
702 bool dtob(const double &val) {
703     if(int(val)>=1)
704         return true;
705     else
706         return false;
707 }
708
709 // integer power function (2^x), which is not available in GCC math library
710 int pow2(const int &val) {
711     int pow = 1;
712     for(int i=0;i<val;i++) {
713         pow *= 2;
714     }
715     return pow;
716 }
717
718 // integer value to bool bits
719 bool itoba(const int &val, const int &width, bool *a) {
720     if(val<pow2(width)) {
721         int rem=val;
722         for(int i=width-1;i>=0;i--) {
723             a[i]=(rem/pow2(i))?true:false;
724             rem=rem%pow2(i);
725         }
726         return 0;                          // succeed
727     }
```

```
728     else
729         return 1;                                   // failed, val is too big
730 }
731
732 /*********************************************************/
733 /*  classes and functions for reading ATP data files  */
734 /*********************************************************/
735
736 enum etype {TR, TL, TC, TLC, TRL, TRC, TRLC, TRLCG,
737             TVsr, TVs, TIs, TSW, TFDTL};            // type of elements
738 enum ecirtype {TBranch, TShunt};                    // type of branch
739 // type of base class
740 enum ecls {TPElement, TAElement, TTLElement, TSWElement};
741
742 // an exception class derived from the class logic_error, which is used for
743 // exception handling in the EMTP program
744 class emtp_error : public logic_error {
745 public:
746     emtp_error (const string &what_arg) : logic_error(what_arg) {}
747 };
748
749 // structure for branch elements (connecting between two nodes)
750 struct CBranch {                                    // branch
751     etype type;                                     // class type of the element
752     ecls cls;                                       // base class type of the element
753     string node1;                                   // Node 1 name
754     string node2;                                   // Node 2 name
755     int pos1;                                       // position index of Node 1
756     int pos2;                                       // position index of Node 2
757     void* data;                                     // object of the element
758 };
759
760 // structure for shunt elements (connecting between one node to ground)
761 struct CShunt {                                     // shunt or load
762     etype type;                                     // class type of the element
763     ecls cls;                                       // base class type of the element
764     string node;                                    // Node name
765     int pos;                                        // position index of Node
766     void* data;                                     // object of the element
767 };
768
769 struct CIout {                                      // current output
770     ecirtype type;                                  // Branch / Shunt
771     int index;                                      // index in branch/shunt vector
772 };
773
774 struct CVout {                                      // voltage output
775     string node;                                    // Node name
776     int index;                                      // index in voltage vector
777 };
778
779 struct CSWNode {                                    // location of switches
780     ecirtype type;                                  // Branch / Shunt
781     int index;                                      // index in branch/shunt vector
782 };
783
784 typedef vector<string> CNVec;
785 typedef vector<CBranch> CBVec;
786 typedef vector<CShunt> CSVec;
787 typedef vector<CIout> CIVec;
788 typedef vector<CVout> CVVec;
789 typedef vector<CSWNode> CSWVec;
790 typedef vector<CMatrix> CMVec;
791
792 typedef vector<CNVec> CModeNVec;
793 typedef vector<CBVec> CModeBVec;
794 typedef vector<CSVec> CModeSVec;
795 typedef vector<CIVec> CModeIVec;
796 typedef vector<CVVec> CModeVVec;
797 typedef vector<CSWVec> CModeSWVec;
798 typedef vector<CMVec> CModeMVec;
799
800 // obtain the G matrix corresponding to the current switch states
801 CMatrix &getCurrentG(CMVec &Gmx, bool *cst, const int &width) {
802     int idx = 0;
803     for(int i=0;i<width;i++) {
804         int tmp = (int) cst[i];
805         int pow2 = 1;
806         for(int j=0;j<i;j++) {
807             pow2 *= 2;
808         }
809         idx += tmp*pow2;
810     }
811     return Gmx[idx];
812 }
813
814 // find position of the nodes in the node vector
815 bool findnode(const CNVec &nodes, const string &node, int &pos) {
816     for(int i=0;i<nodes.size();i++) {
817         if(nodes.at(i) == node) {
818             pos = i;
819             return true;
820         }
821     }
822     return false;
823 }
824
825 // right trim the string if blank characters exist
826 string rtrim(const string &str) {
827     string tstr(str);
828     int len = tstr.length();
829     while (len != 0) {
830         if(tstr[len-1] == ' ') {
831             tstr.erase(len-1,1);
832             len--;
833         }
834         else
835             break;
836     }
837     return tstr;
838 }
839
840 // left trim the string if blank characters exist
841 string ltrim(const string &str) {
842     string tstr(str);
843     int len = tstr.length();
844     while (len != 0) {
845         if(tstr[0] == ' ') {
846             tstr.erase(0,1);
847             len--;
848         }
849         else
850             break;
851     }
852     return tstr;
853 }
854
855 // trim the string if blank characters exist
856 string trim(const string &str) {
857     string tstr(str);
858     return ltrim(rtrim(tstr));
859 }
860
861 // check blank strings
862 bool isblank(const string &str) {
863     string tmpstr = string(str);
864     return (ltrim(tmpstr)).length()?false:true;
865 }
866
867 // swap two strings
868 void swapstr(string &str1, string &str2) {
869     string tmp;
870     tmp = str1;
871     str1 = str2;
872     str2 = tmp;
873 }
```

```
874
875 // make an uppercase copy of s:
876 string uppercase(const string &s) {
877   char* buf = new char[s.length()];
878   s.copy(buf, s.length());
879   for(int i = 0; i < s.length(); i++)
880     buf[i] = toupper(buf[i]);
881   string r(buf, s.length());
882   delete buf;
883   return r;
884 }
885
886 // make a lowercase copy of s:
887 string lowercase(const string &s) {
888   char* buf = new char[s.length()];
889   s.copy(buf, s.length());
890   for(int i = 0; i < s.length(); i++)
891     buf[i] = tolower(buf[i]);
892   string r(buf, s.length());
893   delete buf;
894   return r;
895 }
896
897 /*  read an RLC branch for ATP data file
898     return values:
899         0 --- a RLC branch has been successfully read
900         1 --- end of file
901         2 --- end of branch card
902         3 --- $include encountered
903         4 --- C BEGIN FDNE encountered
904         5 --- BEGIN NEW DATA CASE encountered
905         6 --- C END FDNE encountered
906         7 --- frequency-dependent line card found
907 */
908 int readrlc(const string &fname,              // ATP data file name
909             ifstream &dat,                    // file stream to read file
910             int &linenum,                     // line number counter
911             string &buf,                      // string buffer for the file
912             string &node1,                    // node 1
913             string &node2,                    // node 2
914             double &r,                        // resistance
915             double &l,                        // inductance
916             double &c,                        // capacitance
917             int &output,                      // request for output
918             bool &vintage                     // precision format for RLC
919             ) throw(emtp_error) {
920     while(!getline(dat,buf).eof()) {
921         // In Linux system, reading windows-format text files will have
922         // a '\r' character in the end of string buffer. It has to be
923         // removed in order to execute the program properly.
924         if(buf[buf.size()-1] == '\r') buf = buf.erase(buf.size()-1,1);
925         linenum++;
926         output = 0;
927         if(buf.size()>=80) {
928             if(buf[79] == '1')
929                 output = 1;
930             //else if(buf[79] == '2')         // voltage output in unsupported
931             //    output = 2;
932             buf = buf.substr(0,79);         // truncate
933         }
934         r=0.0;l=0.0;c=0.0;
935         buf = rtrim(buf);
936         int cnt = buf.length();
937         if(cnt==0) continue;                // blank line
938         // compare the first letter
939         if(buf[0]=='C'||buf[0]=='c') {
940             if(uppercase(buf.substr(1,11))==" BEGIN FDNE")
941                 return 4;                   // BEGIN FDNE found
942             else if(uppercase(buf.substr(1,9))==" END FDNE")
943                 return 6;                   // END FDNE found
944             else
945                 continue;                   // comment line
946         }
947         if(buf[0]=='/') return 2;           // end of branch card
948         if(buf[0]=='-') return 7;           // frequency-dependent line found
949         string tmpstr = uppercase(buf.substr(0,5));
950         if(tmpstr=="BEGIN") {
951             if(uppercase(rtrim(buf)) == "BEGIN NEW DATA CASE") {
952                 if(!getline(dat,buf).eof()) {
953                     // find BLANK
954                     if(buf[buf.size()-1] == '\r')
955                         buf = buf.erase(buf.size()-1,1);
956                     linenum++;
957                     if(uppercase(rtrim(buf)) != "BLANK") {
958                         if(buf[0]=='C' || buf[0]=='c')
959                             return 5;   // BEGIN NEW DATA CASE encountered
960                         else {
961                             stringstream errstr;
962                             errstr << fname << " - Invalid data read at line "
963                                 << linenum << " : \"" << buf << "\", "
964                                 << "comment card must be followed with "
965                                 << "BEGIN NEW DATA CASE";
966                             throw emtp_error(errstr.str());
967                         }
968                     }
969                     else
970                         return 1;           // end of file
971                 }
972                 else {
973                     stringstream errstr;
974                     errstr << fname << " - Invalid data read at line "
975                         << linenum << " : \"" << buf << "\"";
976                     throw emtp_error(errstr.str());
977                 }
978             }
979             else
980                 continue;                   // ignore
981         }
982         if(tmpstr=="BLANK") {
983             if(uppercase(rtrim(buf)) == "BLANK")
984                 return 1;                   // end of file
985             else
986                 continue;
987         }
988         if(buf[0]=='$') {                   // ATP variables
989             if(uppercase(buf.substr(1,7))=="INCLUDE")
990                 return 3;                   // $include branches
991             else if(uppercase(buf.substr(1,7))=="VINTAGE") {
992                 string valstr = ltrim(buf.substr(8,buf.size()-8));
993                 if(valstr[0]==',') {
994                     valstr.erase(0,1);
995                     valstr=ltrim(valstr);
996                 }
997                 if(valstr=="1") {
998                     vintage = true;
999                 }
1000                else if(valstr=="0") {
1001                    vintage = false;
1002                }
1003                else {
1004                    stringstream errstr;
1005                    errstr << fname << " - Invalid data read at line "
1006                        << linenum << " : \"" << buf << "\", "
1007                        << "invalid variable assignment";
1008                    throw emtp_error(errstr.str());
1009                }
1010                continue;
1011            }
1012            else
1013                continue;                   // ignore
1014        }
1015        node1 = uppercase(buf.substr(2,6));
1016        node2 = uppercase(buf.substr(8,6));
1017        string resis, induc, capac;
1018        if(vintage) {
1019            if(cnt>=74) {                   // R, L, C
```

```
1020                resis = buf.substr(26,16);
1021                induc = buf.substr(26+16,16);
1022                capac = buf.substr(26+2*16,16);
1023                r = isblank(resis)?0.0:atof(ltrim(resis).c_str());
1024                l = isblank(induc)?0.0:(1.0e-3*atof(ltrim(induc).c_str()));
1025                c = isblank(capac)?0.0:(1.0e-6*atof(ltrim(capac).c_str()));
1026            }
1027            else if(cnt>=58) {                  // R, L
1028                resis = buf.substr(26,16);
1029                induc = buf.substr(26+16,16);
1030                r = isblank(resis)?0.0:atof(ltrim(resis).c_str());
1031                l = isblank(induc)?0.0:(1.0e-3*atof(ltrim(induc).c_str()));
1032            }
1033            else if(cnt>=42) {                  // R
1034                resis = buf.substr(26,16);
1035                r = isblank(resis)?0.0:atof(ltrim(resis).c_str());
1036            }
1037            else {
1038                stringstream errstr;
1039                errstr << fname << " - Invalid data read at line "
1040                    << linenum << " : \"" << buf << "\"";
1041                throw emtp_error(errstr.str());
1042            }
1043        }
1044        else {
1045            if(cnt>=44) {                       // R, L, C
1046                resis = buf.substr(26,6);
1047                induc = buf.substr(26+6,6);
1048                capac = buf.substr(26+2*6,6);
1049                r = isblank(resis)?0.0:atof(ltrim(resis).c_str());
1050                l = isblank(induc)?0.0:(1.0e-3*atof(ltrim(induc).c_str()));
1051                c = isblank(capac)?0.0:(1.0e-6*atof(ltrim(capac).c_str()));
1052            }
1053            else if(cnt>=38) {                  // R, L
1054                resis = buf.substr(26,6);
1055                induc = buf.substr(26+6,6);
1056                r = isblank(resis)?0.0:atof(ltrim(resis).c_str());
1057                l = isblank(induc)?0.0:(1.0e-3*atof(ltrim(induc).c_str()));
1058            }
1059            else if(cnt>=32) {                  // R
1060                resis = buf.substr(26,6);
1061                r = isblank(resis)?0.0:atof(ltrim(resis).c_str());
1062            }
1063            else {
1064                stringstream errstr;
1065                errstr << fname << " - Invalid data read at line "
1066                    << linenum << " : \"" << buf << "\"";
1067                throw emtp_error(errstr.str());
1068            }
1069        }
1070        break;
1071    }
1072    if(dat.eof())
1073        return 1;                               // end of reading data file
1074    else
1075        return 0;
1076 }
1077
1078
1079 int readfdne(const string &fname,              // ATP data file name
1080              ifstream &fdnedat,                // file stream for reading file
1081              string &buf,                      // string buffer for the file
1082              double &dt,                       // time step size
1083              CNVec &nodes,                     // node vector
1084              CBVec &branches,                  // branch vector
1085              CSVec &shunts,                    // shunt vector
1086              bool &vintage,                    // precision format for RLC
1087              int &linenum                      // line number counter
1088              ) throw(emtp_error) {
1089    if(!fdnedat.is_open()) throw emtp_error(fname+" - File open error.");
1090    string cn;
1091    CBranch cb;
1092    CShunt cs;
1093    // read file
1094    string node1, node2, node3, node4, node5, node6;
1095    int output = 0;
1096    double r, l, c, g, tmp1, tmp2;
1097    int pos;
1098    // three types of branches: R, RL, RLCG
1099    while(true) {
1100        int retval = readrlc(fname, fdnedat, linenum, buf, node1, node2,
1101            r, l, c, output, vintage);
1102        if(retval) return retval;
1103        if(!isblank(node1) && !isblank(node2)) {
1104            // possible a branch
1105            if(node1.substr(1,2)=="__" && node2.substr(1,2)=="__") {
1106                // a branch
1107                if(l==0.0) {                    // R
1108                    cb.node1 = node1;
1109                    cb.node2 = node2;
1110                    cb.type = TR;
1111                    cb.cls = TPElement;
1112                    cb.data = new CR(r);
1113                    if(!findnode(nodes, node1, pos)) {
1114                        cn = string(node1);
1115                        nodes.push_back(cn);
1116                        cb.pos1 = nodes.size() - 1;
1117                    }
1118                    else
1119                        cb.pos1 = pos;
1120                    if(!findnode(nodes, node2, pos)) {
1121                        cn = string(node2);
1122                        nodes.push_back(cn);
1123                        cb.pos2 = nodes.size() - 1;
1124                    }
1125                    else
1126                        cb.pos2 = pos;
1127                    branches.push_back(cb);
1128                }
1129                else {                          // RL
1130                    cb.node1 = node1;
1131                    cb.node2 = node2;
1132                    cb.type = TRL;
1133                    cb.cls = TPElement;
1134                    cb.data = new CRL(r, l, dt);
1135                    if(!findnode(nodes, node1, pos)) {
1136                        cn = string(node1);
1137                        nodes.push_back(cn);
1138                        cb.pos1 = nodes.size() - 1;
1139                    }
1140                    else
1141                        cb.pos1 = pos;
1142                    if(!findnode(nodes, node2, pos)) {
1143                        cn = string(node2);
1144                        nodes.push_back(cn);
1145                        cb.pos2 = nodes.size() - 1;
1146                    }
1147                    else
1148                        cb.pos2 = pos;
1149                    branches.push_back(cb);
1150                }
1151            }
1152            else if(node1.substr(1,2)=="__" ||
1153                node2.substr(1,2)=="__") {      // RLCG
1154                // read one more line
1155                bool isbranch = false;
1156                if(!readrlc(fname, fdnedat, linenum, buf, node3, node4,
1157                    g, tmp1, c, output, vintage)) {
1158                    if(isblank(node3) || node3.substr(1,2)=="__")
1159                        swapstr(node3, node4);
1160                    if(g == 0.0) {              // read G
1161                        if(readrlc(fname, fdnedat, linenum, buf, node5, node6,
1162                            g, tmp1, tmp2, output, vintage)) {
1163                            stringstream errstr;
1164                            errstr << fname << " - Invalid data read at line "
1165                                << linenum << " : \"" << buf << "\"";
```

```
                    throw emtp_error(errstr.str());
                }
            }
            else {                          // read C
                if(readric(fname, fdnedat, linenum, buf, node5, node6,
                    tmp1, tmp2, c, output, vintage)) {
                    stringstream errstr;
                    errstr << fname << " - Invalid data read at line "
                        << linenum << " : \"" << buf << "\"";
                    throw emtp_error(errstr.str());
                }
            }
            if(isblank(node5) || node5.substr(1,2)=="__")
                swapstr(node5, node6);
            if(!isblank(node6))         // RLCG branch
                isbranch = true;
            // else RLCG shunt
        }
        if(node4!=node6 || node3!=node5) {
            stringstream errstr;
            errstr << fname << " - Invalid data read at line "
                << linenum << " : \"" << buf << "\"";
            throw emtp_error(errstr.str());
        }
        if(node1 == node3) node1 = node2;
        node2 = node4;
        if(isbranch) {                  // RLCG branch
            cb.node1 = node1;
            cb.node2 = node2;
            cb.type = TRLCG;
            cb.cls = TPElement;
            cb.data = new CRLCG(r, l, c, g, dt);
            if(!findnode(nodes, node1, pos)) {
                cn = string(node1);
                nodes.push_back(cn);
                cb.pos1 = nodes.size() - 1;
            }
            else
                cb.pos1 = pos;
            if(!findnode(nodes, node2, pos)) {
                cn = string(node2);
                nodes.push_back(cn);
                cb.pos2 = nodes.size() - 1;
            }
            else
                cb.pos2 = pos;
            branches.push_back(cb);
        }
        else {                          // RLCG shunt
            cs.node = node1;
            cs.type = TRLCG;
            cs.cls = TPElement;
            cs.data = new CRLCG(r, l, c, g, dt);
            if(!findnode(nodes, node1, pos)) {
                cn = string(node1);
                nodes.push_back(cn);
                cs.pos = nodes.size() - 1;
            }
            else
                cs.pos = pos;
            shunts.push_back(cs);
        }
    }
    else {
        stringstream errstr;
        errstr << fname << " - Invalid data read at line "
            << linenum << " : \"" << buf << "\"";
        throw emtp_error(errstr.str());
    }
}
else {                                  // L
    if(isblank(node1)) node1 = node2;
    if(r == 0.0) {
        cs.node = node1;
        cs.type = TL;
        cs.cls = TPElement;
        cs.data = new CL(l, dt);
        if(!findnode(nodes, node1, pos)) {
            cn = string(node1);
            nodes.push_back(cn);
            cs.pos = nodes.size() - 1;
        }
        else
            cs.pos = pos;
        shunts.push_back(cs);
    }
    else if(l == 0.0) {                 // R
        cs.node = node1;
        cs.type = TR;
        cs.cls = TPElement;
        cs.data = new CR(r);
        if(!findnode(nodes, node1, pos)) {
            cn = string(node1);
            nodes.push_back(cn);
            cs.pos = nodes.size() - 1;
        }
        else
            cs.pos = pos;
        shunts.push_back(cs);
    }
    else {                              // RL
        cs.node = node1;
        cs.type = TRL;
        cs.cls = TPElement;
        cs.data = new CRL(r, l, dt);
        if(!findnode(nodes, node1, pos)) {
            cn = string(node1);
            nodes.push_back(cn);
            cs.pos = nodes.size() - 1;
        }
        else
            cs.pos = pos;
        shunts.push_back(cs);
    }
  }
 }
}

void readfdtlparam(const string &atpdatfn,  // ATP data file name
                   ifstream &atpdat,        // file stream for the file
                   string &buf,             // string buffer for the file
                   const int &N,            // order of Zc/Al
                   double *para,            // parameters
                   int &linenum             // line number counter
                   ) throw(emtp_error) {
    int i, j, k=0;
    for(i=0;i<N/3;i++) {
        if(!getline(atpdat,buf).eof()) {
            if(buf[buf.size()-1] == '\r')
                buf = buf.erase(buf.size()-1,1);
            linenum++;
            buf = rtrim(buf);
            for(j=0;j<3;j++) {
                para[k] = atof(ltrim(buf.substr(j*26,26)).c_str());
                k++;
            }
        }
        else {
            stringstream errstr;
            errstr << atpdatfn << " - Invalid data read at line "
                << linenum << " : \"" << buf << "\"";
            throw emtp_error(errstr.str());
        }
    }
    if(N%3!=0) {
        if(!getline(atpdat,buf).eof()) {
```

```
1312            if(buf[buf.size()-1] == '\r')
1313                buf = buf.erase(buf.size()-1,1);
1314            linenum++;
1315            buf = rtrim(buf);
1316            for(j=0;j<N%3;j++) {
1317                para[k] = atof(ltrim(buf.substr(j*26,26)).c_str());
1318                k++;
1319            }
1320        }
1321        else {
1322            stringstream errstr;
1323            errstr << atpdatfn << " - Invalid data read at line "
1324                << linenum << " : \"" << buf << "\"";
1325            throw emtp_error(errstr.str());
1326        }
1327    }
1328 }
1329
1330 void readfdtl(const string &atpdatfn,    // ATP data file name
1331               ifstream &atpdat,          // file stream for the file
1332               string &buf,               // string buffer for the file
1333               double &dt,                // time step size
1334               CNVec &nodes,              // node vector
1335               CBVec &branches,           // branch vector
1336               CSVec &shunts,             // shunt vector
1337               CIVec &curout,             // current output vector
1338               bool &vintage,             // precision format for RLC
1339               int &linenum               // line number counter
1340               ) throw(emtp_error) {
1341    if(!atpdat.is_open()) throw emtp_error(atpdatfn+" - File open error.");
1342    string cn;
1343    CBranch cb;
1344    CShunt cs;
1345    CIout cio;
1346    // read file
1347    string node1, node2;
1348    int output = 0;
1349    bool isshunt;
1350    int pos;
1351
1352    int NZc, NP;
1353    double Tau, ZcCon;
1354    double *ZcRes, *ZcPol, *PRes, *PPol;
1355
1356    if(buf[1]!='1') {
1357        stringstream errstr;
1358        errstr << atpdatfn << " - Invalid data read at line "
1359            << linenum << " : \"" << buf << "\", "
1360            << "multiphase lines are unsupported.";
1361        throw emtp_error(errstr.str());
1362    }
1363    if(buf.size()>=80) {
1364        if(buf[79] == '1')
1365            output = 1;
1366        buf = buf.substr(0,79);              // truncate
1367    }
1368    else if(rtrim(buf).size()<54) {
1369        stringstream errstr;
1370        errstr << atpdatfn << " - Invalid data read at line "
1371            << linenum << " : \"" << buf << "\"";
1372        throw emtp_error(errstr.str());
1373    }
1374    if(buf[55]!='1') {
1375        stringstream errstr;
1376        errstr << atpdatfn << " - Invalid data read at line "
1377            << linenum << " : \"" << buf << "\", "
1378            << "multiphase lines are unsupported.";
1379        throw emtp_error(errstr.str());
1380    }
1381    node1 = uppercase(buf.substr(2,6));
1382    node2 = uppercase(buf.substr(8,6));
1383    isshunt=false;
1384    if(isblank(node1)) {                     // shunt
1385        swapstr(node1, node2);
1386        isshunt=true;
1387    }
1388    else if(isblank(node2)) {                // still shunt
1389        isshunt=true;
1390    }
1391
1392    // read Zc(s)
1393    if(!getline(atpdat,buf).eof()) {
1394        if(buf[buf.size()-1] == '\r')
1395            buf = buf.erase(buf.size()-1,1);
1396        linenum++;
1397        NZc = atoi(ltrim(buf.substr(0,8)).c_str());
1398        ZcCon = atof(ltrim(buf.substr(8,32)).c_str());
1399    }
1400    else {
1401        stringstream errstr;
1402        errstr << atpdatfn << " - Invalid data read at line "
1403            << linenum << " : \"" << buf << "\"";
1404        throw emtp_error(errstr.str());
1405    }
1406    ZcRes = new double[NZc];
1407    ZcPol = new double[NZc];
1408    readfdtlparam(atpdatfn, atpdat, buf, NZc, ZcRes, linenum);
1409    readfdtlparam(atpdatfn, atpdat, buf, NZc, ZcPol, linenum);
1410
1411    // read A1(s)/P(s)
1412    if(!getline(atpdat,buf).eof()) {
1413        if(buf[buf.size()-1] == '\r')
1414            buf = buf.erase(buf.size()-1,1);
1415        linenum++;
1416        NP = atoi(ltrim(buf.substr(0,8)).c_str());
1417        Tau = atof(ltrim(buf.substr(8,32)).c_str());
1418    }
1419    else {
1420        stringstream errstr;
1421        errstr << atpdatfn << " - Invalid data read at line "
1422            << linenum << " : \"" << buf << "\"";
1423        throw emtp_error(errstr.str());
1424    }
1425    PRes = new double[NP];
1426    PPol = new double[NP];
1427    readfdtlparam(atpdatfn, atpdat, buf, NP, PRes, linenum);
1428    readfdtlparam(atpdatfn, atpdat, buf, NP, PPol, linenum);
1429
1430    if(isshunt) {                            // shunt
1431        cs.node = node1;
1432        cs.type = TFDTL;
1433        cs.cls = TTLElement;
1434        cs.data = new CFdtl(ZcCon, ZcRes, ZcPol, NZc, Tau,
1435            PRes, PPol, NP, dt);
1436        if(!findnode(nodes, node1, pos)) {
1437            cn = string(node1);
1438            nodes.push_back(cn);
1439            cs.pos = nodes.size() - 1;
1440        }
1441        else
1442            cs.pos = pos;
1443        shunts.push_back(cs);
1444        if(output) {
1445            cio.type = TShunt;
1446            cio.index = shunts.size() - 1;
1447            curout.push_back(cio);
1448        }
1449    }
1450    else { // branch
1451        cb.node1 = node1;
1452        cb.node2 = node2;
1453        cb.type = TFDTL;
1454        cb.cls = TTLElement;
1455        cb.data = new CFdtl(ZcCon, ZcRes, ZcPol, NZc, Tau,
1456            PRes, PPol, NP, dt);
1457        if(!findnode(nodes, node1, pos)) {
```

```
1458            cn = string(node1);
1459            nodes.push_back(cn);
1460            cb.pos1 = nodes.size() - 1;
1461        }
1462        else
1463            cb.pos1 = pos;
1464        if(!findnode(nodes, node2, pos)) {
1465            cn = string(node2);
1466            nodes.push_back(cn);
1467            cb.pos2 = nodes.size() - 1;
1468        }
1469        else
1470            cb.pos2 = pos;
1471        branches.push_back(cb);
1472        if(output) {
1473            cio.type = TBranch;
1474            cio.index = branches.size() - 1;
1475            curout.push_back(cio);
1476        }
1477    }
1478    delete []ZcRes;
1479    delete []ZcPol;
1480    delete []PRes;
1481    delete []PPol;
1482    // ignore transformation matrix since this is a single-phase system
1483    for(int i=0;i<2;i++) {
1484        if(!getline(atpdat,buf).eof()) {
1485            linenum++;
1486        }
1487        else {
1488            stringstream errstr;
1489            errstr << atpdatfn << " - Invalid data read at line "
1490                   << linenum << " : \"" << buf << "\"";
1491            throw emtp_error(errstr.str());
1492        }
1493    }
1494 }
1495
1496 int readbranchcard(const string &atpdatfn,  // ATP data file name
1497                    ifstream &atpdat,        // file stream for reading file
1498                    string &buf,             // string buffer for the file
1499                    double &dt,              // time step size
1500                    CNVec &nodes,            // node vector
1501                    CBVec &branches,         // branch vector
1502                    CSVec &shunts,           // shunt vector
1503                    CIVec &curout,           // current output vector
1504                    int &linenum,            // line number counter
1505                    string &newcs            // new sorting card found
1506                    ) throw(emtp_error) {
1507    string cn;
1508    CBranch cb;
1509    CShunt cs;
1510    CIout cio;
1511    // read file
1512    string node1, node2;
1513    int output;
1514    double r, l, c;
1515    int retval, pos;
1516    bool searchfdne=false, isshunt, vintage=false;
1517    stringstream errstr;
1518    while(true) {
1519    retval=readric(atpdatfn, atpdat, linenum, buf, node1, node2,
1520        r, l, c, output, vintage);
1521    switch(retval) {
1522    case 0:                                 // succeed
1523        isshunt=false;
1524        if(isblank(node1)) {                // shunt
1525            swapstr(node1, node2);
1526            isshunt=true;
1527        }
1528        else if(isblank(node2)) {
1529            isshunt=true;
1530        }
1531
1532        if(r==0.0) {                        // R=0
1533            if(l==0.0) {                    // C only
1534                if(isshunt) {               // shunt
1535                    cs.node = node1;
1536                    cs.type = TC;
1537                    cs.cls = TPElement;
1538                    cs.data = new CC(c, dt);
1539                    if(!findnode(nodes, node1, pos)) {
1540                        cn = string(node1);
1541                        nodes.push_back(cn);
1542                        cs.pos = nodes.size() - 1;
1543                    }
1544                    else
1545                        cs.pos = pos;
1546                    shunts.push_back(cs);
1547                    if(output) {
1548                        cio.type = TShunt;
1549                        cio.index = shunts.size() - 1;
1550                        curout.push_back(cio);
1551                    }
1552                }
1553                else {                      // branch
1554                    cb.node1 = node1;
1555                    cb.node2 = node2;
1556                    cb.type = TC;
1557                    cb.cls = TPElement;
1558                    cb.data = new CC(c, dt);
1559                    if(!findnode(nodes, node1, pos)) {
1560                        cn = string(node1);
1561                        nodes.push_back(cn);
1562                        cb.pos1 = nodes.size() - 1;
1563                    }
1564                    else
1565                        cb.pos1 = pos;
1566                    if(!findnode(nodes, node2, pos)) {
1567                        cn = string(node2);
1568                        nodes.push_back(cn);
1569                        cb.pos2 = nodes.size() - 1;
1570                    }
1571                    else
1572                        cb.pos2 = pos;
1573                    branches.push_back(cb);
1574                    if(output) {
1575                        cio.type = TBranch;
1576                        cio.index = branches.size() - 1;
1577                        curout.push_back(cio);
1578                    }
1579                }
1580            }
1581            else if(c==0.0) {               // L only
1582                if(isshunt) {               // shunt
1583                    cs.node = node1;
1584                    cs.type = TL;
1585                    cs.cls = TPElement;
1586                    cs.data = new CL(l, dt);
1587                    if(!findnode(nodes, node1, pos)) {
1588                        cn = string(node1);
1589                        nodes.push_back(cn);
1590                        cs.pos = nodes.size() - 1;
1591                    }
1592                    else
1593                        cs.pos = pos;
1594                    shunts.push_back(cs);
1595                    if(output) {
1596                        cio.type = TShunt;
1597                        cio.index = shunts.size() - 1;
1598                        curout.push_back(cio);
1599                    }
1600                }
1601                else {                      // branch
1602                    cb.node1 = node1;
1603                    cb.node2 = node2;
```

```
                cb.type = TL;
                cb.cls = TPElement;
                cb.data = new CL(l, dt);
                if(!findnode(nodes, node1, pos)) {
                    cn = string(node1);
                    nodes.push_back(cn);
                    cb.pos1 = nodes.size() - 1;
                }
                else
                    cb.pos1 = pos;
                if(!findnode(nodes, node2, pos)) {
                    cn = string(node2);
                    nodes.push_back(cn);
                    cb.pos2 = nodes.size() - 1;
                }
                else
                    cb.pos2 = pos;
                branches.push_back(cb);
                if(output) {
                    cio.type = TBranch;
                    cio.index = branches.size() - 1;
                    curout.push_back(cio);
                }
            }
        }
        else {                          // LC
            if(isshunt) {               // shunt
                cs.node = node1;
                cs.type = TLC;
                cs.cls = TPElement;
                cs.data = new CLC(l, c, dt);
                if(!findnode(nodes, node1, pos)) {
                    cn = string(node1);
                    nodes.push_back(cn);
                    cs.pos = nodes.size() - 1;
                }
                else
                    cs.pos = pos;
                shunts.push_back(cs);
                if(output) {
                    cio.type = TShunt;
                    cio.index = shunts.size() - 1;
                    curout.push_back(cio);
                }
            }
            else {                      // branch
                cb.node1 = node1;
                cb.node2 = node2;
                cb.type = TLC;
                cb.cls = TPElement;
                cb.data = new CLC(l, c, dt);
                if(!findnode(nodes, node1, pos)) {
                    cn = string(node1);
                    nodes.push_back(cn);
                    cb.pos1 = nodes.size() - 1;
                }
                else
                    cb.pos1 = pos;
                if(!findnode(nodes, node2, pos)) {
                    cn = string(node2);
                    nodes.push_back(cn);
                    cb.pos2 = nodes.size() - 1;
                }
                else
                    cb.pos2 = pos;
                branches.push_back(cb);
                if(output) {
                    cio.type = TBranch;
                    cio.index = branches.size() - 1;
                    curout.push_back(cio);
                }
            }
        }
    }
    else {                              // R not equal to 0
        if(l==0.0) {                    // RC or R branch
            if(c==0.0) {                // R
                if(isshunt) {           // shunt
                    cs.node = node1;
                    cs.type = TR;
                    cs.cls = TPElement;
                    cs.data = new CR(r);
                    if(!findnode(nodes, node1, pos)) {
                        cn = string(node1);
                        nodes.push_back(cn);
                        cs.pos = nodes.size() - 1;
                    }
                    else
                        cs.pos = pos;
                    shunts.push_back(cs);
                    if(output) {
                        cio.type = TShunt;
                        cio.index = shunts.size() - 1;
                        curout.push_back(cio);
                    }
                }
                else {                  // branch
                    cb.node1 = node1;
                    cb.node2 = node2;
                    cb.type = TR;
                    cb.cls = TPElement;
                    cb.data = new CR(r);
                    if(!findnode(nodes, node1, pos)) {
                        cn = string(node1);
                        nodes.push_back(cn);
                        cb.pos1 = nodes.size() - 1;
                    }
                    else
                        cb.pos1 = pos;
                    if(!findnode(nodes, node2, pos)) {
                        cn = string(node2);
                        nodes.push_back(cn);
                        cb.pos2 = nodes.size() - 1;
                    }
                    else
                        cb.pos2 = pos;
                    branches.push_back(cb);
                    if(output) {
                        cio.type = TBranch;
                        cio.index = branches.size() - 1;
                        curout.push_back(cio);
                    }
                }
            }
            else {                      // RC
                if(isshunt) {           // shunt
                    cs.node = node1;
                    cs.type = TRC;
                    cs.cls = TPElement;
                    cs.data = new CRC(r, c, dt);
                    if(!findnode(nodes, node1, pos)) {
                        cn = string(node1);
                        nodes.push_back(cn);
                        cs.pos = nodes.size() - 1;
                    }
                    else
                        cs.pos = pos;
                    shunts.push_back(cs);
                    if(output) {
                        cio.type = TShunt;
                        cio.index = shunts.size() - 1;
                        curout.push_back(cio);
                    }
                }
                else {                  // branch
                    cb.node1 = node1;
```

```
                cb.node2 = node2;
                cb.type = TRC;
                cb.cls = TPElement;
                cb.data = new CRC(r, c, dt);
                if(!findnode(nodes, node1, pos)) {
                    cn = string(node1);
                    nodes.push_back(cn);
                    cb.pos1 = nodes.size() - 1;
                }
                else
                    cb.pos1 = pos;
                if(!findnode(nodes, node2, pos)) {
                    cn = string(node2);
                    nodes.push_back(cn);
                    cb.pos2 = nodes.size() - 1;
                }
                else
                    cb.pos2 = pos;
                branches.push_back(cb);
                if(output) {
                    cio.type = TBranch;
                    cio.index = branches.size() - 1;
                    curout.push_back(cio);
                }
            }
        }
    }
    else if(c==0.0) {                   // RL
        if(isshunt) {                   // shunt
            cs.node = node1;
            cs.type = TRL;
            cs.cls = TPElement;
            cs.data = new CRL(r, l, dt);
            if(!findnode(nodes, node1, pos)) {
                cn = string(node1);
                nodes.push_back(cn);
                cs.pos = nodes.size() - 1;
            }
            else
                cs.pos = pos;
            shunts.push_back(cs);
            if(output) {
                cio.type = TShunt;
                cio.index = shunts.size() - 1;
                curout.push_back(cio);
            }
        }
        else {                          // branch
            cb.node1 = node1;
            cb.node2 = node2;
            cb.type = TRL;
            cb.cls = TPElement;
            cb.data = new CRL(r, l, dt);
            if(!findnode(nodes, node1, pos)) {
                cn = string(node1);
                nodes.push_back(cn);
                cb.pos1 = nodes.size() - 1;
            }
            else
                cb.pos1 = pos;
            if(!findnode(nodes, node2, pos)) {
                cn = string(node2);
                nodes.push_back(cn);
                cb.pos2 = nodes.size() - 1;
            }
            else
                cb.pos2 = pos;
            branches.push_back(cb);
            if(output) {
                cio.type = TBranch;
                cio.index = branches.size() - 1;
                curout.push_back(cio);
            }
```

```
            }
        }
        else {                          // RLC
            if(isshunt) {               // shunt
                cs.node = node1;
                cs.type = TRLC;
                cs.cls = TPElement;
                cs.data = new CRLC(r, l, c, dt);
                if(!findnode(nodes, node1, pos)) {
                    cn = string(node1);
                    nodes.push_back(cn);
                    cs.pos = nodes.size() - 1;
                }
                else
                    cs.pos = pos;
                shunts.push_back(cs);
                if(output) {
                    cio.type = TShunt;
                    cio.index = shunts.size() - 1;
                    curout.push_back(cio);
                }
            }
            else {                      // branch
                cb.node1 = node1;
                cb.node2 = node2;
                cb.type = TRLC;
                cb.cls = TPElement;
                cb.data = new CRLC(r, l, c, dt);
                if(!findnode(nodes, node1, pos)) {
                    cn = string(node1);
                    nodes.push_back(cn);
                    cb.pos1 = nodes.size() - 1;
                }
                else
                    cb.pos1 = pos;
                if(!findnode(nodes, node2, pos)) {
                    cn = string(node2);
                    nodes.push_back(cn);
                    cb.pos2 = nodes.size() - 1;
                }
                else
                    cb.pos2 = pos;
                branches.push_back(cb);
                if(output) {
                    cio.type = TBranch;
                    cio.index = branches.size() - 1;
                    curout.push_back(cio);
                }
            }
        }
    }
    break;
case 1:                                 // end of file
    return 1;
case 2:                                 // end of branch card
    newcs = buf.substr(1,6);
    return 0;
case 3: // $include encountered
        // FDTL
        //readfdtl function
        // this functionality is not incorporated into this program
        // all frequency-dependent line model data must be included
        // in the main ATP data file.
    break;
case 4:                                 // C BEGIN FDNE encountered
    if(readfdne(stpdatfn, stpdat, buf, dt, nodes, branches, shunts,
        vintage, linenum) == 3) {
        string fdnefn=trim(buf.substr(8,buf.size()-8));
        if(fdnefn[0]==',') {
            fdnefn.erase(0,1);
            fdnefn=ltrim(fdnefn);
        }
        // read FDNE data file
```

```
            ifstream fdnedat(fdnefn.c_str());
            if(!fdnedat.is_open()) {
                throw emtp_error(fdnefn+" - File open error.");
            }
            int fdnelm=0;
            string fdnebuf;
            int val = readfdne(fdnefn, fdnedat, fdnebuf, dt, nodes, branches,
                shunts, vintage, fdnelm);
        }
        break;
    case 5:                                 // BEGIN NEW DATA CASE encountered
        newcs = "BNDC";
        return 1;
    case 6:                                 // C END FDNE encountered
        if(searchfdne) searchfdne = false;
        break;
    case 7: // fdt1 card found
        readfdt1(atpdatfn, atpdat, buf, dt, nodes, branches, shunts, curout,
            vintage, linenum);
        break;
    default:
        errstr << atpdatfn << " - Invalid data read at line "
            << linenum << " : \"" << buf << "\"";
        throw emtp_error(errstr.str());
    }
    }
}

int readswitchcard(const string &atpdatfn,  // ATP data file name
                   ifstream &atpdat,        // file stream for reading file
                   string &buf,             // string buffer for the file
                   double &dt,              // time step size
                   CNVec &nodes,            // node vector
                   CBVec &branches,         // branch vector
                   CSVec &shunts,           // shunt vector
                   CIVec &curout,           // current output vector
                   CSWVec &sw,              // switch vector
                   int &linenum,            // line number counter
                   string &newcs            // sorting branch card found
                   ) throw(emtp_error) {
    string cn;
    CBranch cb;
    CShunt cs;
    CIout cio;
    CSWNode swnode;
    // read file
    string node1, node2;
    int output, pos;
    bool isshunt;
    while(!getline(atpdat,buf).eof()) {
        if(buf[buf.size()-1] == '\r')
            buf = buf.erase(buf.size()-1,1);
        linenum++;
        output = 0;
        if(buf.size()>=80) {
            if(buf[79] == '1')
                output = 1;
            //else if(buf[79] == '2')       // voltage output is unsupported
            //  output = 2;
            buf = buf.substr(0,79);         // truncate
        }
        buf = rtrim(buf);
        int cnt = buf.length();
        if(cnt==0) continue;                // blank line;
        // reading first letter
        if(buf[0]=='C'||buf[0]=='c')
            continue;                       // comment line
        if(buf[0]=='/') {
            newcs = buf.substr(1,6);
            return 0;                       // end of branch card
        }
        string tmpstr = uppercase(buf.substr(0,5));
        if(tmpstr=="BEGIN") {
            if(uppercase(rtrim(buf)) == "BEGIN NEW DATA CASE") {
                if(!getline(atpdat,buf).eof()) {
                    // find BLANK
                    if(buf[buf.size()-1] == '\r')
                        buf = buf.erase(buf.size()-1,1);
                    linenum++;
                    if(uppercase(rtrim(buf)) != "BLANK") {
                        if(buf[0]=='C' || buf[0]=='c')
                            newcs = "BNDC"; // BEGIN NEW DATA CASE encountered
                        else {
                            stringstream errstr;
                            errstr << atpdatfn
                                << " - Invalid data read at line "
                                << linenum << " : \"" << buf << "\". "
                                << "comment card must be followed with "
                                << "BEGIN NEW DATA CASE";
                            throw emtp_error(errstr.str());
                        }
                    }
                    return 1;
                }
                else {
                    stringstream errstr;
                    errstr << atpdatfn << " - Invalid data read at line "
                        << linenum << " : \"" << buf << "\"";
                    throw emtp_error(errstr.str());
                }
            }
            else
                continue;                   // ignore
        }
        if(tmpstr=="BLANK") {
            if(uppercase(rtrim(buf)) == "BLANK")
                return 1;                   // end of file
            else
                continue;
        }
        if(buf[0]=='$') {                   // ATP variables
            continue;                       // ignore
        }
        node1 = uppercase(buf.substr(2,6));
        node2 = uppercase(buf.substr(8,6));
        isshunt=false;
        if(isblank(node1)) {                // shunt
            swapstr(node1, node2);
            isshunt=true;
        }
        else if(isblank(node2)) {
            isshunt=true;
        }
        if(isshunt) {                       // shunt
            cs.node = node1;
            cs.type = TSW;
            cs.cls = TSWElement;
            cs.data = new CSwitch;

            if(!findnode(nodes, node1, pos)) {
                cn = string(node1);
                nodes.push_back(cn);
                cs.pos = nodes.size() - 1;
            }
            else
                cs.pos = pos;
            shunts.push_back(cs);

            swnode.type = TShunt;
            swnode.index = shunts.size() - 1;
            sw.push_back(swnode);

            if(output) {
                cio.type = TShunt;
                cio.index = shunts.size() - 1;
                curout.push_back(cio);
```

```
2042            }
2043        }
2044        else {                                  // branch
2045            cb.node1 = node1;
2046            cb.node2 = node2;
2047            cb.type = TSW;
2048            cb.cls = TSWElement;
2049            cb.data = new CSwitch;
2050
2051            if(!findnode(nodes, node1, pos)) {
2052                cn = string(node1);
2053                nodes.push_back(cn);
2054                cb.pos1 = nodes.size() - 1;
2055            }
2056            else
2057                cb.pos1 = pos;
2058            if(!findnode(nodes, node2, pos)) {
2059                cn = string(node2);
2060                nodes.push_back(cn);
2061                cb.pos2 = nodes.size() - 1;
2062            }
2063            else
2064                cb.pos2 = pos;
2065            branches.push_back(cb);
2066
2067            swnode.type = TBranch;
2068            swnode.index = branches.size() - 1;
2069            sw.push_back(swnode);
2070
2071            if(output) {
2072                cio.type = TBranch;
2073                cio.index = branches.size() - 1;
2074                curout.push_back(cio);
2075            }
2076        }
2077    }
2078    return 1;                                   // eof found
2079
2080 }
2081
2082 int readoutputcard(const string &atpdatfn,  // ATP data file name
2083                    ifstream &atpdat,        // file stream for reading file
2084                    string &buf,             // string buffer for the file
2085                    double &dt,              // time step size
2086                    CNVec &nodes,            // node vector
2087                    CBVec &branches,         // branch vector
2088                    CSVec &shunts,           // shunt vector
2089                    CVVec &volout,           // voltage output vector
2090                    int &linenum,            // line number counter
2091                    string &newcs            // new sorting card found
2092                    ) throw(emtp_error) {
2093    CVout vio;
2094    // read file
2095    string node;
2096    while(!getline(atpdat,buf).eof()) {
2097        if(buf[buf.size()-1] == '\r') buf = buf.erase(buf.size()-1,1);
2098        linenum++;
2099        buf = rtrim(buf);
2100        int cnt = buf.length();
2101        if(cnt==0) continue;                    // blank line;
2102        // first letter
2103        if(buf[0]=='C'||buf[0]=='c')
2104            continue;                           // comment line
2105        if(buf[0]=='/') {
2106            newcs = buf.substr(1,6);
2107            return 0;                           // end of branch card
2108        }
2109        string tmpstr = uppercase(buf.substr(0,5));
2110        if(tmpstr=="BEGIN") {
2111            if(uppercase(rtrim(buf)) == "BEGIN NEW DATA CASE") {
2112                if(!getline(atpdat,buf).eof()) {
2113                    // find BLANK
2114                    if(buf[buf.size()-1] == '\r')
2115                        buf = buf.erase(buf.size()-1,1);
2116                    linenum++;
2117                    if(uppercase(rtrim(buf)) != "BLANK") {
2118                        if(buf[0]=='C' || buf[0]=='c')
2119                            newcs = "BNDC"; // BEGIN NEW DATA CASE encountered
2120                        else {
2121                            stringstream errstr;
2122                            errstr << atpdatfn
2123                                << " - Invalid data read at line "
2124                                << linenum << " : \"" << buf << "\", "
2125                                << "comment card must be followed with "
2126                                << "BEGIN NEW DATA CASE";
2127                            throw emtp_error(errstr.str());
2128                        }
2129                    }
2130                    return 1;
2131                }
2132                else {
2133                    stringstream errstr;
2134                    errstr << atpdatfn << " - Invalid data read at line "
2135                        << linenum << " : \"" << buf << "\"";
2136                    throw emtp_error(errstr.str());
2137                }
2138            }
2139            else
2140                continue;                       // ignore
2141        }
2142        if(tmpstr=="BLANK") {
2143            if(uppercase(rtrim(buf)) == "BLANK")
2144                return 1;                       // end of file
2145            else
2146                continue;
2147        }
2148        if(buf[0]=='$') {                       // ATP variables
2149            continue;                           // ignore
2150        }
2151        buf = ltrim(buf);
2152        cnt = buf.size();
2153        if(cnt%6!=0) {
2154            stringstream errstr;
2155            errstr << atpdatfn << " - Invalid data read at line "
2156                << linenum << " : \"" << buf << "\"";
2157            throw emtp_error(errstr.str());
2158        }
2159        for(int i=0;i<cnt/6;i++) {
2160            node = uppercase(buf.substr(6*i,6));
2161            int pos;
2162            if(!findnode(nodes, node, pos)) {
2163                stringstream errstr;
2164                errstr << atpdatfn << " - Invalid data read at line "
2165                    << linenum << " : \"" << buf << "\", "
2166                    << "output node not exist.";
2167                throw emtp_error(errstr.str());
2168            }
2169            else {
2170                vio.node = node;
2171                vio.index = pos;
2172                volout.push_back(vio);
2173            }
2174        }
2175    }
2176    return 1;                                   // eof found
2177 }
2178
2179 int readsourcecard(const string &atpdatfn,  // ATP data file name
2180                    ifstream &atpdat,        // file stream for reading file
2181                    string &buf,             // string buffer for the file
2182                    double &dt,              // time step size
2183                    CNVec &nodes,            // node vector
2184                    CBVec &branches,         // branch vector
2185                    CSVec &shunts,           // shunt vector
2186                    CIVec &curout,           // current output vector
2187                    int &linenum,            // line number counter
```

```
                    string &newcs           // new sorting card found
                    ) throw(emtp_error) {
string cn;
CShunt cs;
// read file
string node;
int pos;
while(!getline(atpdat,buf).eof()) {
    if(buf[buf.size()-1] == '\r')
        buf = buf.erase(buf.size()-1,1);
    linenum++;
    if(buf.size()>=80) {
        buf = buf.substr(0,40);         // truncate
    }
    buf = rtrim(buf);
    int cnt = buf.length();
    if(cnt==0) continue;                // blank line;
    // first letter
    if(buf[0]=='C'||buf[0]=='c')
        continue;                       // comment line
    if(buf[0]=='/') {
        newcs = buf.substr(1,6);
        return 0;                       // end of branch card
    }
    string tmpstr = uppercase(buf.substr(0,5));
    if(tmpstr=="BEGIN") {
        if(uppercase(rtrim(buf)) == "BEGIN NEW DATA CASE") {
            if(!getline(atpdat,buf).eof()) {
                // find BLANK
                if(buf[buf.size()-1] == '\r')
                    buf = buf.erase(buf.size()-1,1);
                linenum++;
                if(uppercase(rtrim(buf)) != "BLANK") {
                    if(buf[0]=='C' || buf[0]=='c')
                        newcs = "BNDC"; // BEGIN NEW DATA CASE encountered
                    else {
                        stringstream errstr;
                        errstr << atpdatfn
                            << " - Invalid data read at line "
                            << linenum << " : \"" << buf << "\", "
                            << "comment card must be foillowed with "
                            << "BEGIN NEW DATA CASE";
                        throw emtp_error(errstr.str());
                    }
                }
                return 1;
            }
            else {
                stringstream errstr;
                errstr << atpdatfn << " - Invalid data read at line "
                    << linenum << " : \"" << buf << "\"";
                throw emtp_error(errstr.str());
            }
        }
        else
            continue;                   // ignore
    }
    if(tmpstr=="BLANK") {
        if(uppercase(rtrim(buf)) == "BLANK")
            return 1;                   // end of file
        else
            continue;
    }
    if(buf[0]=='$') {                   // ATP variables
        continue;                       // ignore
    }
    if(buf.substr(0,2)!="14") {
        stringstream errstr;
        errstr << atpdatfn << " - Invalid data read at line "
            << linenum << " : \"" << buf << "\", "
            << "only voltage/current source type 14 supported";
        throw emtp_error(errstr.str());
    }
    node = uppercase(buf.substr(2,6));
    double mag = atof(ltrim(buf.substr(10,10)).c_str());
    double f = atof(ltrim(buf.substr(20,10)).c_str());
    double pha = atof(ltrim(buf.substr(30,10)).c_str());
    cs.node = node;
    if(buf.substr(8,2)=="-1") {         // current source
        cs.type = TIs;
        cs.data = new CIs(mag, f, pha);
    }
    else {                              // voltage source
        cs.type = TVs;
        cs.data = new CVs(mag, f, pha);
    }
    cs.cls = TAElement;
    if(!findnode(nodes, node, pos)) {
        cn = string(node);
        nodes.push_back(cn);
        cs.pos = nodes.size() - 1;
    }
    else
        cs.pos = pos;
    shunts.push_back(cs);
}
return 1;                               // eof encountered
}

int readdatacase(const string &atpdatfn,    // ATP data file name
                 ifstream &atpdat,          // file stream for reading file
                 double &dt,                // time step size
                 double &f,                 // power frequency
                 CNVec &nodes,              // node vector
                 CBVec &branches,           // branch vector
                 CSVec &shunts,             // shunt vector
                 CIVec &curout,             // current output vector
                 CVVec &volout,             // voltage output vector
                 CSWVec &sw,                // switch vector
                 bool &searchdt,            // search dt or not
                 const bool &readdt,        // read dt or not
                 int &linenum
                 ) throw(emtp_error) {
    string buf, newcs;
    while(!getline(atpdat,buf).eof()) {
        if(buf[buf.size()-1] == '\r') buf = buf.erase(buf.size()-1,1);
        linenum++;
        buf = rtrim(buf);
        int cnt = buf.length();
        if(cnt==0) continue;                // blank line;
        // first letter
        if(buf[0]=='C'||buf[0]=='c')
            continue;                       // comment line
        if(buf[0]=='$') {                   // ATP variables
            continue;                       // ignore
        }
        if(cnt < 5) {
            stringstream errstr;
            errstr << atpdatfn << " - Invalid data read at line "
                << linenum << " : \"" << buf << "\"";
            throw emtp_error(errstr.str());
        }
        string tmpstr = uppercase(buf.substr(0,5));
        if(tmpstr=="BEGIN") {
            if(uppercase(rtrim(buf)) == "BEGIN NEW DATA CASE") {
                if(!getline(atpdat,buf).eof()) {
                    // find BLANK
                    if(buf[buf.size()-1] == '\r')
                        buf = buf.erase(buf.size()-1,1);
                    linenum++;
                    if(uppercase(rtrim(buf)) == "BLANK")
                        return 1;           // end of file
                    else {
                        if(buf[0]=='C' || buf[0]=='c') {
                            searchdt = true;
                            continue;
```

```
                }
                else {
                    stringstream errstr;
                    errstr << atpdatfn
                        << " - Invalid data read at line "
                        << linenum << " : \"" << buf << "\", "
                        << "comment card must be followed with "
                        << "BEGIN NEW DATA CASE";
                    throw emtp_error(errstr.str());
                }
            }
        }
        else {
            stringstream errstr;
            errstr << atpdatfn << " - Invalid data read at line "
                << linenum << " : \"" << buf << "\"";
            throw emtp_error(errstr.str());
        }
    }
    continue;
}
if(tmpstr=="BLANK") {
    if(uppercase(rtrim(buf)) == "BLANK")
        return 1;                       // end of file
    else
        continue;
}
if(uppercase(buf.substr(0,15))=="POWER FREQUENCY") {
    // read power frequency f
    f = atof(ltrim(buf.substr(31,8)).c_str());
    continue;
}
if(searchdt) {
    // currently, the support of XOPT and COPT
    // is not incorporated into this program
    if(readdt) dt = atof(ltrim(buf.substr(0,8)).c_str());
    if(!getline(atpdat,buf).eof()) { // other settings are ignored
        if(buf[buf.size()-1] == '\r')
            buf = buf.erase(buf.size()-1,1);
        linenum++;
        searchdt = false;
        continue;
    }
    else {
        stringstream errstr;
        errstr << atpdatfn << " - Invalid data read at line "
            << linenum << " : \"" << buf << "\"";
        throw emtp_error(errstr.str());
    }
}

if(buf[0]=='/') {                       // beginning of cards
    if(dt == 0.0) {
        stringstream errstr;
        errstr << atpdatfn << " - Invalid data read at line "
            << linenum << " : \"" << buf << "\","
            << "simulation time-step is undefined.";
        throw emtp_error(errstr.str());
    }
    string cardspec=uppercase(buf.substr(1,6));
    while(true) {
        if(cardspec=="BRANCH") {
            // reading branches
            if(readbranchcard(atpdatfn, atpdat, buf, dt,
                nodes, branches, shunts,
                curout, linenum, newcs)) {
                if(newcs=="BNDC") {
                    searchdt = true;
                    return 0;
                }
                else
                    return 1;
            }
            cardspec = newcs;
        }
        else if(cardspec=="SWITCH") {
            // reading switches
            if(readswitchcard(atpdatfn, atpdat, buf, dt,
                nodes, branches, shunts,
                curout, sw, linenum, newcs)) {
                if(newcs=="BNDC") {
                    searchdt = true;
                    return 0;
                }
                else
                    return 1;
            }
            cardspec = newcs;
        }
        else if(cardspec=="OUTPUT") {
            // reading outputs
            if(readoutputcard(atpdatfn, atpdat, buf, dt,
                nodes, branches, shunts,
                volout, linenum, newcs)) {
                if(newcs=="BNDC") {
                    searchdt = true;
                    return 0;
                }
                else
                    return 1;
            }
            cardspec = newcs;
        }
        else if(cardspec=="SOURCE") {
            // reading sources
            if(readsourcecard(atpdatfn, atpdat, buf, dt,
                nodes, branches, shunts,
                curout, linenum, newcs)) {
                if(newcs=="BNDC") {
                    searchdt = true;
                    return 0;
                }
                else
                    return 1;
            }
            cardspec = newcs;
        }
        else {
            stringstream errstr;
            errstr << atpdatfn << " - Invalid data read at line "
                << linenum << " : \"" << buf << "\"";
            throw emtp_error(errstr.str());
        }
    }
}
return 1;                               // end of file
}

#endif
```

E.2 emtp.cpp

```
/*  File    : emtp.cpp
 *  Abstract:
 *
 *      C++ implementation of SimuLINK S-function
 *      for Real-time EMTP
 *
 *  by Xin Nie, Power Engineering Group
 *  Dept. of Electrical and Computer Engineering
```

```
 *  University of Alberta, Edmonton, Canada
 *  email: xnie@ece.ualberta.ca
 *  May 10, 2005
 *
 *  This program is based on a C++ S-function template,
 *  which is the copyright of Mathworks Inc.
 *
 */

/*  this section of code is used to compile this program in
    Visual C++. It must be disabled when compile in MATLAB  */
/*#if (_MSC_VER <= 1200)  // including MSVC 6.0
    #pragma warning(disable:4786)
    #ifndef MATLAB_MEX_FILE
        #define MATLAB_MEX_FILE
    #endif
#endif*/

// defines max input and output port width for S-function block
#define MAXINPUTPORTWIDTH          3
#define MAXOUTPUTPORTWIDTH         3

#include <fstream>
#include <string>
#include <sstream>
#include <vector>
#include "matrix.h"
#include "emtp.h"

using namespace std;
using namespace math;

/*****************************************/
/*    definition of static variables     */
/*****************************************/

// simulation variables
static double dt;                          // simulation time-step size
static string atpdatfn;                    // ATP data file name
static int InputPortWidth;                 // input port width
static int OutputPortWidth;                // output port width

// simulation objects to store network elements
static CModeNVec md_nodes;
static CModeBVec md_branches;
static CModeSVec md_shunts;
static CModeIVec md_curout;
static CModeVVec md_volout;
static CModeSWVec md_sw;
static CModeMVec md_Gmx;
static CMVec md_vnode;
static vector<bool> SFInput;
static vector<double> SFOutput;

void EMTPInitialize(CModeNVec &md_nodes, CModeBVec &md_branches,
                    CModeSVec &md_shunts, CModeIVec &md_curout,
                    CModeVVec &md_volout, CModeSWVec &md_sw,
                    CModeMVec &md_Gmx, CMVec &md_vnode,
                    const string &atpdatfn, double& f, double& dt,
                    const bool& readdt, int& InputPortWidth,
                    int& OutputPortWidth) throw(exception);

void EMTPUpdate(CModeNVec &md_nodes, CModeBVec &md_branches,
                CModeSVec &md_shunts, CModeIVec &md_curout,
                CModeVVec &md_volout, CModeSWVec &md_sw,
                CModeMVec &md_Gmx, CMVec &md_vnode,
                const string &atpdatfn, double& t, vector<bool> &SFInput,
                int& InputPortWidth, vector<double> &SFOutput,
                int& OutputPortWidth) throw(exception);

void EMTPTerminate(CModeBVec &md_branches, CModeSVec &md_shunts)
                throw(exception);

#ifdef __cplusplus
extern "C" { // use the C fcn-call standard for all functions
#endif       // defined within this scope

#define S_FUNCTION_LEVEL 2
#define S_FUNCTION_NAME  emtp

/*
 * Need to include simstruc.h for the definition of the SimStruct and
 * its associated macro definitions.
 */
#include "simstruc.h"

/*====================*
 * S-function methods *
 *====================*/

/* Function: mdlInitializeSizes ===============================================
 * Abstract:
 *    The sizes information is used by Simulink to determine the S-function
 *    block's characteristics (number of inputs, outputs, states, etc.).
 */
static void mdlInitializeSizes(SimStruct *S)
{
    /* See sfuntmpl.doc for more details on the macros below */

    int_T NInputPort = 1;
    int_T NOutputPort = 1;
    int_T NSampleTime = 1;

    ssSetNumSFcnParams(S, 2);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        /* Return if number of expected != number of actual parameters */
        return;
    }

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, NInputPort)) return;
    ssSetInputPortWidth(S, 0, MAXINPUTPORTWIDTH);

    if (!ssSetNumOutputPorts(S, NOutputPort)) return;
    ssSetOutputPortWidth(S, 0, MAXOUTPUTPORTWIDTH);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    ssSetNumSampleTimes(S, NSampleTime);
    ssSetNumRWork(S, 0);                   // reserve for real numbers
    ssSetNumIWork(S, 0);                   // reserve for integers
    ssSetNumPWork(S, 10);                  // reserve for pointers
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    ssSetOptions(S, 0);
}

/* Function: mdlInitializeSampleTimes =========================================
 * Abstract:
 *    This function is used to specify the sample time(s) for your
 *    S-function. You must register the same number of sample times as
 *    specified in ssSetNumSampleTimes.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, mxGetScalar(ssGetSFcnParam(S, 0)));
    ssSetOffsetTime(S, 0, 0.0);
}

// #undef MDL_START
#define MDL_START  /* Change to #undef to remove function */
#if defined(MDL_START)
  /* Function: mdlStart =======================================================
```

```
155     * Abstract:
156     *    This function is called once at start of model execution. If you
157     *    have states that should be initialized once, this is the place
158     *    to do it.
159     */
160     static void mdlStart(SimStruct *S)
161     {
162         // get simulation time-step from s-function parameters
163
164         dt = (double) mxGetScalar(ssGetSFcnParam(S, 0));
165         double f = 0.0;
166         bool readdt = false;
167         const mxArray *para2 = ssGetSFcnParam(S, 1);
168         int len = (mxGetM(para2) * mxGetN(para2)) + 1;
169         char *str = new char[len];
170         if(mxGetString(para2, str, len)) {
171             static const char *msg = "Parameter #2 - string is truncated.";
172             ssSetErrorStatus(S, msg);
173         }
174         atpdatfn = trim(string(str));        // ATP data file
175         delete []str;
176
177         md_nodes.clear();
178         md_branches.clear();
179         md_shunts.clear();
180         md_curout.clear();
181         md_volout.clear();
182         md_sw.clear();
183         md_Gmx.clear();
184
185         try {
186             EMTPInitialize(md_nodes, md_branches, md_shunts, md_curout,
187                 md_volout, md_sw, md_Gmx, md_vnode, atpdatfn, f, dt, readdt,
188                 InputPortWidth, OutputPortWidth);
189         }
190         catch(exception &e) {
191             // cout << e.what() << endl;
192             // send the error message to Simulink
193             static const char *msg = e.what();
194             ssSetErrorStatus(S, msg);
195         }
196
197         // Initialize vectors for reading inputs and outputs
198         SFInput.clear();
199         int i;
200         for(i=0;i<InputPortWidth;i++) {
201             SFInput.push_back((bool)0);
202         }
203         SFOutput.clear();
204         for(i=0;i<OutputPortWidth;i++) {
205             SFOutput.push_back((double)0.0);
206         }
207     }
208 #endif /*  MDL_START */
209
210 /* Function: mdlOutputs =======================================================
211  * Abstract:
212  *    In this function, you compute the outputs of your S-function
213  *    block. Generally outputs are placed in the output vector, ssGetY(S).
214  */
215 static void mdlOutputs(SimStruct *S, int_T tid)
216 {
217     // get input
218     InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
219     // get output
220     real_T            *y = ssGetOutputPortRealSignal(S,0);
221
222     // update input
223     int i;
224     for(i=0;i<InputPortWidth;i++) {
225         SFInput.at(i) = !(dtob(*uPtrs[i]));
226     }
227
228     // update output
229     for(i=0;i<OutputPortWidth;i++) {
230         y[i] = SFOutput.at(i);
231     }
232
233 }
234
235 //#undef MDL_UPDATE
236 #define MDL_UPDATE  /* Change to #undef to remove function */
237 #if defined(MDL_UPDATE)
238     /* Function: mdlUpdate ======================================================
239      * Abstract:
240      *    This function is called once for every major integration time step.
241      *    Discrete states are typically updated here, but this function is
242      *    useful for performing any tasks that should only take place once per
243      *    integration step.
244      */
245     static void mdlUpdate(SimStruct *S, int_T tid)
246     {
247         // get current simulation time and time step
248         double t = ssGetT(S);
249
250         try {
251             EMTPUpdate(md_nodes, md_branches, md_shunts, md_curout, md_volout,
252                 md_sw, md_Gmx, md_vnode, atpdatfn, t, SFInput, InputPortWidth,
253                 SFOutput, OutputPortWidth);
254         }
255         catch(exception &e) {
256             // cout << e.what() << endl;
257             // send the error message to Simulink
258             static const char *msg = e.what();
259             ssSetErrorStatus(S, msg);
260         }
261     }
262 #endif /* MDL_UPDATE */
263
264 /* Function: mdlTerminate =====================================================
265  * Abstract:
266  *    In this function, you should perform any.actions that are necessary
267  *    at the termination of a simulation.  For example, if memory was
268  *    allocated in mdlStart, this is the place to free it.
269  */
270 static void mdlTerminate(SimStruct *S)
271 {
272     try {
273         EMTPTerminate(md_branches, md_shunts);
274     }
275     catch(exception &e) {
276         // cout << e.what() << endl;
277         // send the error message to Simulink
278         static const char *msg = e.what();
279         ssSetErrorStatus(S, msg);
280     }
281 }
282 /*============================================================*
283  * See sfuntmpl.doc for the optional S-function methods *
284  *============================================================*/
285
286 /*============================*
287  * Required S-function trailer *
288  *============================*/
289
290 #ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
291 #include "simulink.c"      /* MEX-file interface mechanism */
292 #else
293 #include "cg_sfun.h"       /* Code generation registration function */
294 #endif
295
296 #ifdef __cplusplus
297 } // end of extern "C" scope
298 #endif
299
300 /*========================*/
```

```
/*   Main EMTP routines   */
/*************************/

void EMTPInitialize(CModeNVec &md_nodes,     // node vector for data cases
                    CModeBVec &md_branches,  // branch vector for data cases
                    CModeSVec &md_shunts,    // shunt vector for data cases
                    CModeIVec &md_curout,    // current output for data cases
                    CModeVVec &md_volout,    // voltage output for data cases
                    CModeSWVec &md_sw,       // switch vector for data cases
                    CModeMVec &md_Gmx,       // system conductance matrix
                    CMVec &md_vnode,         // system voltage vector
                    const string &atpdatfn,  // ATP data file name
                    double& f,               // power frequency
                    double& dt,              // time step size
                    const bool& readdt,      // whether read dt in data file
                    int& InputPortWidth,     // input port width
                    int& OutputPortWidth     // output port width
                    ) throw(exception) {
    ifstream atpdat(atpdatfn.c_str());
    if(!atpdat.is_open())
        throw emtp_error(atpdatfn+" - File open error.");
    bool searchdt = false;
    int linenum = 0;                         // line number counter
    int nsw = 0;                             // number of switch states
    int i, k;
    int ipw = 0, opw = 0;

    while(true) {
        CNVec nodes;
        CBVec branches;
        CSVec shunts;
        CIVec curout;
        CVVec volout;
        CSWVec sw;
        CMVec Gmx;
        nodes.clear();
        branches.clear();
        shunts.clear();
        curout.clear();
        volout.clear();
        sw.clear();
        Gmx.clear();

        double fs = 0.0;

        // read ATP data file
        int retval = readdatacase(atpdatfn, atpdat, dt, fs,
            nodes, branches, shunts, curout, volout, sw,
            searchdt, readdt, linenum);

        int Nnode = nodes.size();        // number of nodes in network

        md_nodes.push_back(nodes);
        md_branches.push_back(branches);
        md_shunts.push_back(shunts);
        md_curout.push_back(curout);
        md_volout.push_back(volout);
        md_sw.push_back(sw);

        // node voltage vector
        CMatrix vnode = CMatrix(Nnode, 1);
        // initialize to 0 to all elements
        int ii, jj;
        for(ii=0;ii<Nnode;ii++) {
            vnode(ii,0) = 0.0;
        }

        md_vnode.push_back(vnode);       // save it

        int nswt = sw.size();
        if(nsw != 0 && nswt != nsw) {
            stringstream errstr;
            errstr << atpdatfn
                << " - number of switch states mismatch.";
            throw emtp_error(errstr.str());
        }
        else
            nsw = nswt;

        if(int(fs)!=0) {
            if(int(f)==0) {
                f = fs;
            }
            else {
                if(int(f)!=int(fs)) {
                    stringstream errstr;
                    errstr << atpdatfn
                        << " - power frequency mismatch.";
                    throw emtp_error(errstr.str());
                }
            }
        }

        CMatrix Gc(Nnode, Nnode);        // conductance matrix
        for(ii=0;ii<Nnode;ii++) {        // initialize to 0
            for(jj=0;jj<Nnode;jj++) {
                Gc(ii,jj) = 0.0;
            }
        }

        /*********************************/
        /*  filling the matrix elements  */
        /*********************************/

        // iterate all branches
        for(i=0;i<branches.size();i++) {
            CBranch &cb = branches.at(i);
            int pos1, pos2;
            pos1 = cb.pos1;
            pos2 = cb.pos2;
            double cg;
            switch(cb.cls) {
                case TPElement:
                    cg = ((CPElement *) cb.data)->getG();
                    Gc(pos1, pos1) += cg;
                    Gc(pos2, pos2) += cg;
                    Gc(pos1, pos2) += -cg;
                    Gc(pos2, pos1) += -cg;
                    break;
                case TAElement:
                    cg = ((CAElement *) cb.data)->getG();
                    Gc(pos1, pos1) += cg;
                    Gc(pos2, pos2) += cg;
                    Gc(pos1, pos2) += -cg;
                    Gc(pos2, pos1) += -cg;
                    break;
                case TTLElement:
                    cg = ((CTLElement *) cb.data)->getG();
                    Gc(pos1, pos1) += cg;
                    Gc(pos2, pos2) += cg;
                    break;
                case TSWElement:
                    break;
                default:
                    stringstream errstr;
                    errstr << atpdatfn << " - Invalid cls at branch No. "
                        << i <<" .";
                    throw emtp_error(errstr.str());
            }
        }

        // iterate all shunts
        for(i=0;i<shunts.size();i++) {
            CShunt &cs = shunts.at(i);
            int pos;
            pos = cs.pos;
```

```
447                 double cg;
448                 switch(cs.cls) {
449                     case TPElement:
450                         cg = ((CPElement *) cs.data)->getG();
451                         Gc(pos, pos) += cg;
452                         break;
453                     case TAElement:
454                         cg = ((CAElement *) cs.data)->getG();
455                         Gc(pos, pos) += cg;
456                         break;
457                     case TTLElement:
458                         cg = ((CTLElement *) cs.data)->getG();
459                         Gc(pos, pos) += cg;
460                         break;
461                     case TSWElement:
462                         break;
463                     default:
464                         stringstream errstr;
465                         errstr << atpdatfn << " - Invalid cls at branch No. "
466                                << i <<".";
467                         throw emtp_error(errstr.str());
468                 }
469             }
470
471             // find G matrices based on different switching states
472             bool *st = new bool[nsw];
473             for(k=0;k<pow2(nsw);k++) {
474                 itoba(k, nsw, st);
475                 CMatrix Gnodei = CMatrix(Nnode, Nnode);
476                 Gnodei = Gc;
477
478                 // iterate all switches
479                 for(i=0;i<nsw;i++) {
480                     CSWNode &cswn = sw.at(i);
481                     if(cswn.type == TBranch) {
482                         // TBranch, branch element
483                         int pos1, pos2;
484                         CBranch &cb = branches.at(cswn.index);
485                         pos1 = cb.pos1;
486                         pos2 = cb.pos2;
487                         double cg = getSWG(st[i]);
488                         Gnodei(pos1, pos1) += cg;
489                         Gnodei(pos2, pos2) += cg;
490                         Gnodei(pos1, pos2) += -cg;
491                         Gnodei(pos2, pos1) += -cg;
492                     }
493                     else {
494                         // TShunt, shunt element
495                         int pos;
496                         CShunt &cs = shunts.at(cswn.index);
497                         pos = cs.pos;
498                         double cg = getSWG(st[i]);
499                         Gnodei(pos, pos) += cg;
500                     }
501                 }
502
503                 Gnodei = !Gnodei;           // find inverse
504                 Gmx.push_back(Gnodei);      // save the matrix
505             }
506
507             md_Gmx.push_back(Gmx);          // save the matrix for data case
508
509             delete []st;
510
511             ipw += sw.size();
512             opw += curout.size() + volout.size();
513
514             if(retval) break;               // all data cases end
515         }
516
517         if(ipw > MAXINPUTPORTWIDTH) {
518             stringstream errstr;
519             errstr << atpdatfn
520                    << " - Input port width exceeds port width limit, "
521                    << "increase MAXINPUTPORTWIDTH.";
522                 throw emtp_error(errstr.str());
523             }
524             else if(opw > MAXOUTPUTPORTWIDTH) {
525                 stringstream errstr;
526                 errstr << atpdatfn
527                     << " - Output port width exceeds port width limit, "
528                     << "increase MAXOUTPUTPORTWIDTH.";
529                 throw emtp_error(errstr.str());
530             }
531             else {
532                 InputPortWidth = ipw;
533                 OutputPortWidth = opw;
534             }
535 }
536
537 void EMTPUpdate(CModeNVec &md_nodes,        // node vector for data cases
538                 CModeBVec &md_branches,     // branch vector for data cases
539                 CModeSVec &md_shunts,       // shunt vector for data cases
540                 CModeIVec &md_curout,       // current output for data cases
541                 CModeVVec &md_volout,       // voltage output for data cases
542                 CModeSWVec &md_sw,          // switch vector for data cases
543                 CModeMVec &md_Gmx,          // system conductance matrix
544                 CMVec &md_vnode,            // system voltage vector
545                 const string &atpdatfn,     // ATP data file name
546                 double& t,                  // current simulation time
547                 vector<bool> &SFInput,      // input vector
548                 int& InputPortWidth,        // input port width
549                 vector<double> &SFOutput,   // output vector
550                 int& OutputPortWidth        // output port width
551                 ) throw(exception)
552 {
553     for(int k=0;k<md_nodes.size();k++) {
554         CNVec &nodes = md_nodes.at(k);
555         CBVec &branches = md_branches.at(k);
556         CSVec &shunts = md_shunts.at(k);
557         CIVec &curout = md_curout.at(k);
558         CVVec &volout = md_volout.at(k);
559         CSWVec &sw = md_sw.at(k);
560         CMVec &Gmx = md_Gmx.at(k);
561         CMatrix &vnode = md_vnode.at(k);
562         int i, j=0, m=0;
563
564         /*************************/
565         /*  update switch states  */
566         /*************************/
567
568         int nsw = sw.size();
569         bool *st = new bool[nsw];
570         int Nnode = nodes.size();
571
572         // convert input to bool types, since the inputs are switch states
573         for(i=0;i<nsw;i++) st[i] = SFInput.at(i);
574
575         // iterate all switches
576         for(i=0;i<sw.size();i++) {
577             CSWNode &cswn = sw.at(i);
578             if(cswn.type == TBranch) {
579                 CBranch &cb = branches.at(cswn.index);
580                 ((CSWElement *) cb.data)->update(st[i]);
581             }
582             else {                              // TShunt
583                 CShunt &cs = shunts.at(cswn.index);
584                 ((CSWElement *) cs.data)->update(st[i]);
585             }
586         }
587         m += nsw;
588
589         CMatrix Gnodei = getCurrentG(Gmx, st, nsw);
590
591         delete []st;
592
```

```
593         /*************************************/
594         /*  filling current source vector I  */
595         /*************************************/
596
597         // construct Inode vector
598         CMatrix Inode = CMatrix(Nnode, 1);
599         for(int ii=0;ii<Nnode;ii++) {
600             Inode(ii,0) = 0.0;
601         }
602
603         // iterate all branches
604         for(i=0;i<branches.size();i++) {
605             CBranch &cb = branches.at(i);
606             int pos1, pos2;
607             pos1 = cb.pos1;
608             pos2 = cb.pos2;
609             double cih;
610             double vk = vnode(pos1,0), vm = vnode(pos2,0);
611             double vdiff = vk - vm;
612             switch(cb.cls) {
613                 case TPElement:
614                     cih = ((CPElement *) cb.data)->getIh();
615                     Inode(pos1, 0) += cih;
616                     Inode(pos2, 0) += -cih;
617                     break;
618                 case TAElement:
619                     cih = ((CAElement *) cb.data)->getIeq(t);
620                     Inode(pos1, 0) += cih;
621                     Inode(pos2, 0) += -cih;
622                     break;
623                 case TTLElement:
624                     if(cb.type == TFDTL) {
625                         ((CFdtl *) cb.data)->updateBkm();
626                     }
627                     cih = ((CTLElement *) cb.data)->getIhk();
628                     Inode(pos1, 0) += cih;
629                     cih = ((CTLElement *) cb.data)->getIhm();
630                     Inode(pos2, 0) += cih;
631                     break;
632                 case TSWElement:
633                     break;
634                 default:
635                     stringstream errstr;
636                     errstr << atpdatfn << " - Invalid cls at branch No. "
637                         << i <<" .";
638                     throw emtp_error(errstr.str());
639             }
640         }
641
642         // iterate all shunts
643         for(i=0;i<shunts.size();i++) {
644             CShunt &cs = shunts.at(i);
645             int pos;
646             pos = cs.pos;
647             double cih;
648             double tmp = 0.0;
649             switch(cs.cls) {
650                 case TPElement:
651                     cih = ((CPElement *) cs.data)->getIh();
652                     Inode(pos, 0) += cih;
653                     break;
654                 case TAElement:
655                     cih = ((CAElement *) cs.data)->getIeq(t);
656                     Inode(pos, 0) += cih;
657                     break;
658                 case TTLElement:
659                     if(cs.type == TFDTL) {
660                         ((CFdti *) cs.data)->updateBkm();
661                     }
662                     cih = ((CTLElement *) cs.data)->getIhk();
663                     Inode(pos, 0) += cih;
664                     break;
665                 case TSWElement:
666                     break;
667                 default:
668                     stringstream errstr;
669                     errstr << atpdatfn << " - Invalid cls at branch No. "
670                         << i <<" .";
671                     throw emtp_error(errstr.str());
672             }
673         }
674
675         /*************************************/
676         /*  solve the equation v=G^(-1)*I    */
677         /*  to obtain node voltage vector    */
678         /*************************************/
679         vnode = Gnodei * Inode;
680
681         /*************************************/
682         /*      update histor terms of       */
683         /*        all network elements       */
684         /*************************************/
685
686         // iterate all branches
687         for(i=0;i<branches.size();i++) {
688             CBranch &cb = branches.at(i);
689             int pos1, pos2;
690             pos1 = cb.pos1;
691             pos2 = cb.pos2;
692             double vk = vnode(pos1,0), vm = vnode(pos2,0);
693             double vdiff = vk - vm;
694             switch(cb.cls) {
695                 case TPElement:
696                     ((CPElement *) cb.data)->update(vdiff);
697                     break;
698                 case TAElement:
699                     break;
700                 case TTLElement:
701                     ((CTLElement *) cb.data)->update(vk, vm);
702                     break;
703                 case TSWElement:
704                     break;
705                 default:
706                     stringstream errstr;
707                     errstr << atpdatfn << " - Invalid cls at branch No. "
708                         << i <<" .";
709                     throw emtp_error(errstr.str());
710             }
711         }
712
713         // iterate all shunts
714         for(i=0;i<shunts.size();i++) {
715             CShunt &cs = shunts.at(i);
716             int pos;
717             pos = cs.pos;
718             double tmp = 0.0;
719             switch(cs.cls) {
720                 case TPElement:
721                     ((CPElement *) cs.data)->update(vnode(pos,0));
722                     break;
723                 case TAElement:
724                     break;
725                 case TTLElement:
726                     ((CTLElement *) cs.data)->update(vnode(pos,0), tmp);
727                     break;
728                 case TSWElement:
729                     break;
730                 default:
731                     stringstream errstr;
732                     errstr << atpdatfn << " - Invalid cls at branch No. "
733                         << i <<" .";
734                     throw emtp_error(errstr.str());
735             }
736         }
737
738         /*************************************/
```

```
739         /* setup current and voltage output*/
740         /*********************************/
741
742         // current outputs
743         for(i=0;i<curout.size();i++) {
744             CIout &cio = curout.at(i);
745             if(cio.type == TBranch) {        // Branch
746                 int pos1, pos2;
747                 CBranch &cb = branches.at(cio.index);
748                 pos1 = cb.pos1;
749                 pos2 = cb.pos2;
750                 double vdiff = vnode(pos1,0)-vnode(pos2,0);
751                 switch(cb.cls) {
752                     case TPElement:
753                         SFOutput.at(j) =
754                             ((CPElement *) cb.data)->getib();
755                         break;
756                     case TAElement:
757                         SFOutput.at(j) =
758                             ((CAElement *) cb.data)->getib(vdiff, t);
759                         break;
760                     case TTLElement:
761                         SFOutput.at(j) =
762                             ((CTLElement *) cb.data)->getik();
763                         break;
764                     case TSWElement:
765                         SFOutput.at(j) =
766                             ((CSWElement *) cb.data)->getib(vdiff);
767                         break;
768                     default:
769                         stringstream errstr;
770                         errstr << atpdatfn << " - Invalid cls at branch No. "
771                             << i <<" .";
772                         throw emtp_error(errstr.str());
773                 }
774                 j++;
775             }
776             else {                           // TShunt
777                 int pos;
778                 CShunt &cs = shunts.at(cio.index);
779                 pos = cs.pos;
780                 switch(cs.cls) {
781                     case TPElement:
782                         SFOutput.at(j) =
783                             ((CPElement *) cs.data)->getib();
784                         break;
785                     case TAElement:
786                         SFOutput.at(j) =
787                             ((CAElement *) cs.data)->getib(vnode(pos,0), t);
788                         break;
789                     case TTLElement:
790                         SFOutput.at(j) =
791                             ((CTLElement *) cs.data)->getik();
792                         break;
793                     case TSWElement:
794                         SFOutput.at(j) =
795                             ((CSWElement *) cs.data)->getib(vnode(pos,0));
796                         break;
797                     default:
798                         stringstream errstr;
799                         errstr << atpdatfn << " - Invalid cls at branch No. "
800                             << i <<" .";
801                         throw emtp_error(errstr.str());
802                 }
803                 j++;
804             }
805         }                                    // end for (2) loop
806         // voltage outputs
807         for(i=0;i<volout.size();i++) {
808             CVout &vio = volout.at(i);
809             SFOutput.at(j) = vnode(vio.index, 0);
810             j++;
811         }
812     }
813 }
814
815 void EMTPTerminate(CModeBVec &md_branches,  // branch vector for data cases
816                    CModeSVec &md_shunts     // shunt vector for data cases
817                    ) throw(exception)
818 {
819     int i=0, k=0;
820     for(k=0;k<md_nodes.size();k++) {
821         CBVec &branches = md_branches.at(k);
822         CSVec &shunts = md_shunts.at(k);
823
824         // delete all branches
825         for(i=0;i<branches.size();i++) {
826             CBranch &cb = branches.at(i);
827             switch(cb.type) {
828                 case TR:
829                     delete ((CR *) cb.data);
830                     break;
831                 case TL:
832                     delete ((CL *) cb.data);
833                     break;
834                 case TC:
835                     delete ((CC *) cb.data);
836                     break;
837                 case TLC:
838                     delete ((CLC *) cb.data);
839                     break;
840                 case TRL:
841                     delete ((CRL *) cb.data);
842                     break;
843                 case TRC:
844                     delete ((CRC *) cb.data);
845                     break;
846                 case TRLC:
847                     delete ((CRLC *) cb.data);
848                     break;
849                 case TRLCG:
850                     delete ((CRLCG *) cb.data);
851                     break;
852                 case TVsr:
853                     delete ((CVsr *) cb.data);
854                     break;
855                 case TVs:
856                     delete ((CVs *) cb.data);
857                     break;
858                 case TIs:
859                     delete ((CIs *) cb.data);
860                     break;
861                 case TSW:
862                     delete ((CSwitch *) cb.data);
863                     break;
864                 case TFDTL:
865                     delete ((CFdtl *) cb.data);
866                     break;
867                 default:;
868             }
869         }
870
871         // delete all shunts
872         for(i=0;i<shunts.size();i++) {
873             CShunt &cs = shunts.at(i);
874             switch(cs.type) {
875                 case TR:
876                     delete ((CR *) cs.data);
877                     break;
878                 case TL:
879                     delete ((CL *) cs.data);
880                     break;
881                 case TC:
882                     delete ((CC *) cs.data);
883                     break;
884                 case TLC:
```

```
            delete ((CLC *) cs.data);
            break;
        case TRL:
            delete ((CRL *) cs.data);
            break;
        case TRC:
            delete ((CRC *) cs.data);
            break;
        case TRLC:
            delete ((CRLC *) cs.data);
            break;
        case TRLCG:
            delete ((CRLCG *) cs.data);
            break;
        case TVar:
            delete ((CVar *) cs.data);
            break;
        case TVs:
            delete ((CVs *) cs.data);
            break;
        case TIs:
            delete ((CIs *) cs.data);
            break;
        case TSW:
            delete ((CSwitch *) cs.data);
            break;
        case TFDTL:
            delete ((CFdtl *) cs.data);
            break;
        default:;
    }
}
```